Multiscale Modelling of Flowing Soft Matter: Copolymers and Emulsions


Doctoral Dissertation
Doctoral Program in Chemical Enginering (31.th cycle)

Multiscale Modelling of Flowing Soft Matter

Copolymers and Emulsions

Hermes Droghetti
* * * * * *

Supervisors
Prof. Daniele Marchisio, Supervisor
Prof. Pietro Asinari, Co-supervisor

Doctoral Examination Committee:
Prof. Wioletta Podgórska, Referee, Warsaw University of Technology
Prof. Gaetano D'Avino, Referee, University of Naples Federico II

Politecnico di Torino
April 30, 2019

I hereby declare that, the contents and organisation of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

. . . . . .

Hermes Droghetti
Turin, April 30, 2019

Summary

This PhD Thesis is about the in-silico simulation of soft matter. A multiscale approach was used to face the current limitations related to the modelling of both complex mixtures at molecular level, and the industrial equipment, needed for their manufacturing, at macroscopic level. The term soft matter refers to those products that are obtained by mixing two or more phases (miscible or immiscible, w/ or w/o surfactants), that flow as a response to the application of a reasonable shear stress. A fundamental aspect, when dealing with soft matter, is related to its rheology. Microstructures, such as micelles and droplets, may influence the viscosity of the final products. These structures can evolve, due to the presence of high shear regions in mixing devices, that influence aggregation, shape changes and re-orientation of the aggregates. The capability of predicting both the final shape and the polydispersity of these peculiar structures is required to overcome limitations of the current empirical models that fail beyond their limits of applicability. The range of applications of soft matter is massive. Home and personal care, food industry, drug delivery are some of the possible applications. In particular, copolymers and emulsions, have been investigated using meso- and macro- modelling, keeping in mind that approaches and systems may be interchanged. Computer simulations have been performed on two different scales, i.e. mesoscale and macroscale, by using respectively Dissipative Particle Dynamics (DPD) and Computational Fluid Dynamics (CFD). Copolymers, e.g. tri-block copolymer (Pluronics by BASF) and water systems have been investigated in both equilibrium and non-equilibrium configurations using DPD, a mesoscale approach. In DPD, molecular identity is lost, and atoms, molecules and particles are replaced by clusters of entities that interact via soft potentials. Equilibrium simulations were performed on LAMMPS for three different species of Pluronics, and phase diagrams, at equilibrium, were validated against experimental results. The microstructures were qualitatively and, when possible, quantitatively validated by identifying their status of aggregation. In particular, when micelles and worm-like structures were present, their sphericity was calculated with an in-house developed clustering algorithm that is able to identify different clusters, calculate their gyration radius and how they evolve during the simulation time in response to shear stress.Tests were repeated for Pluronics P104 and P85, proving that the set of simulation parameters ensures the scalability of the models on similar mixtures. This reduces the necessity of performing lab-scale experiments to identify peculiar microstructures.

Shear effects, e.g. because of the stirring, have been evaluated, at this scale, by looking at the modification of the microphases, such as coalescence, deformation, and alignment. In order to perform these simulations, Lees-Edwards Boundary Conditions (LEBC) were used meaning that a linear velocity profile is obtained across the simulation box, such that the viscosity of the system can be obtained from the stress tensor, directly obtained via DPD.

Attention was focused on the variation of the number of aggregates identified, their morphological changes, and the variation of the viscosity in time for all the range of concentration. Shear thinning behavior, observed between 40% and 65% vol. of Pluronics was confirmed with experimental tests, while Newtonian behavior lasts at low concentration.

On the other side, in the manufacturing process to produce commercial soft matter products, stirring is a fundamental operation. In this work, mixing at industrial scale was simulated by using Computational Fluid Dynamics (CFD). Two mixers, a stirred tank with inclined impeller and anchor and an inline rotor-stator mixer, were simulated by using the two-fluid model and population balance equation. Emulsions of silicone-oil in water, at low volume fraction (1% vol.) and different viscosities, and surfactants were investigated on a full 3D scale, with in-house implementation of population balance equation into Ansys Fluent. Multiple reference framework was used to reproduce the stirring effect and results have been validated against the empirical correlations for the power number of these systems and experimental evolution of the droplet size distribution.

In order to obtain the evolution of droplet size distribution for all the silicone-oil-in-water mixtures, two kernels have been implemented in Fluent and tested to better describe the breakage phenomenon, while coalescence was turned off because of the presence of surfactant into the system.

In this work we explored the capabilities of computer simulations of assessing both the quasi-molecular and the macro scale, regarding the manufacturing of products related to home and personal care industries. It was proved that coupling simulations with experiments is a powerful tool that can be used to speed up and optimize industrial processes. This will reduce the gap of knowledge about the many aspects of soft matter, such as all those phenomena that can only be appreciated at molecular level. Eventually, it was developed an OpenFOAM solver (C++, CFD code, open source code), where CFD scale sends information, such as shear rate and concentration of components, to DPD scale and receives a feedback on the evolution of transport coefficients. However, the conversion between the values obtained from DPD and real quantities is still missing. In the future, this tool will allow 3D macroscale simulations where models are directly derived from mesoscale observations.

Acknowledgements

And I would like to acknowledge Professor Ignacio Pagonabarraga, Professor Paola Carbone and Principal Engineer Adam Kowalski.

I would like to dedicate this thesis to my loving parents

Contents

List of Tables

List of Figures

Chapter 1

Thesis Guidelines

1.1 Current Limitations

In the personal and home care industries, complex fluids are widely used for many different applications, and depending on some specific physical properties, they may determine the success or failure of a brand in the market. Complex fluids are manufactured by mixing numerous ingredients including polymers, copolymers, ionic and non-ionic surfactants, and water in large mixing vessels. Although the manufacturing process might seem simple at first sight, the inherent complexity of such systems, in turn caused by the high number but also variety of microscopic molecular structures and patterns, that can be obtained by varying formulations and operating conditions, make the use of predictive computational models extremely useful. Such models become interesting in scaling up/down equipment from lab to pilot and industrial scale, or for process optimization.

An additional reason that makes computational models quite interesting is that dealing with complex fluids from an experimental point of view is quite challenging because of the high number and nature of interactions between the many ingredients. Even though they are fundamental, experiments may be not sufficient to overcome uncertainties and often, their high cost, the difficulties associated in construction of pilot plants, the lack of standardization and difficulties in reproducing similar operating conditions, makes the understanding of the process by experiments above challenging

On the other side, computational models can help scientists and engineers in better understanding the physics and chemistry behind each experiment and provide the necessary support to empirical observations. Different length and time scales, meaning different geometrical and temporal ranges, can be assessed by different computational models. In fact, the complexity of those systems may require a multiscale approach that involves different levels of resolution.

Complex fluids, and in particular those belonging to the category of ”soft matter”, have in common some peculiar features, such as the presence of microstructures that drive most of the macroscopic features of the final products. In this work we are mainly

concerned in emulsions, whose microstructure is constituted by droplets, and structured fluids, whose microstructure is constituted by molecular clusters and assembly, resulting in spherical, rod-like micelles and lamellar structures. In both systems, the main microscopic feature induced by a specific microstructure is the final rheology. Particular attention is paid, in this work, on aggregation and breakage of micellar structures, in structured fluids or droplets in emulsions.

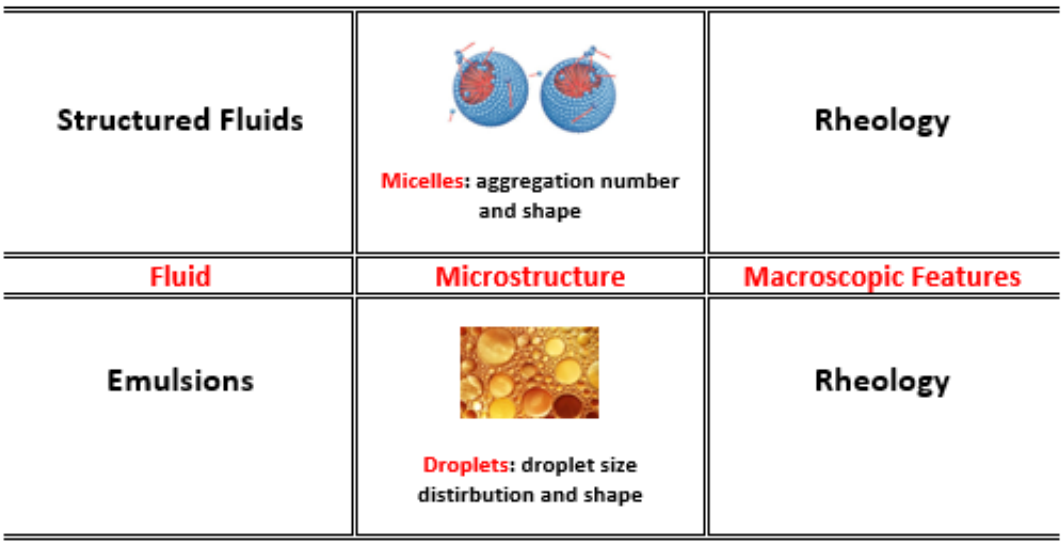


Figure 1.1. *A graphical summary of the different aspects covered in this thesis: two different systems have been assessed from a microscopical point of view to determine macroscopic properties by using computer simulations.*

The current state of the development of the computational models that are currently used to describe these systems at the macroscale, are mostly based on empirical evidences such that they behave properly only in limited ranges of applications. They may therefore produce extremely erroneous outcomes where extrapolated or when wrong assumptions are made, or even worse, when the underlying physics and chemistry are ignored. Microscale molecular models are capable of extract information from a semi-atomistic level of resolution, but they become too computationally expensive when it comes to explore intermediate scales. This is why coarse-grained models were developed, however still now their application is limited because of many uncertainties that come with them. They are considered as toy models that do not provide useful information.

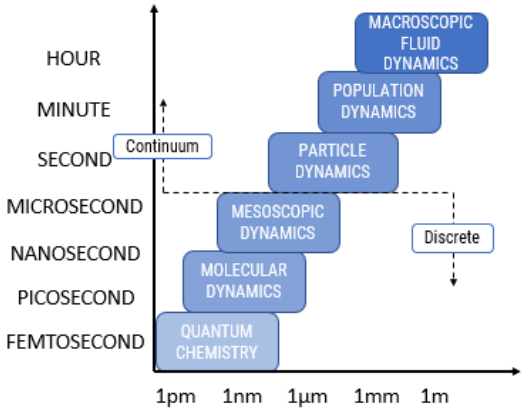


Figure 1.2. *Modelling scales are reported as a function of length and time.*

It is important to highlight that different scales need different models to be reproduced. Atomistic models are used at the molecular scale, mesoscopic models are used at mesoscale and continuum models are used at macroscopic (or continuum) scale. Different models have completely different hypothesis and ways of being applied. In this work, the principal scales will be assessed and the models explained in details.

1.2 Objective of the thesis

The main objective of this thesis is to provide a guide on how chemical computational models may be used in industrial applications where the complexity of involved products and processes is extremely high. In these specific cases, the deeper the level of investigation that may be achieved, the better the results that can be provided. For these industries where products have such low added value, and so many different versions are released every year, the success is based on peculiar properties that can be manipulated and finely tuned only through complex formulations. Wastes must be reduced as much as possible, and experiments can be coupled to predictions from computational models in a smart way, to help forecasting those properties. In this work, two different scales are addressed by using computational models. In fact, the complexity of the phenomena involved in the fate of the microstructures present in structured fluids and emulsions, can be treated with the help of computational models. Structured fluids and emulsions are assessed with two approaches that are capable of describing these systems as continua and as composed by discrete elements. In particular, structured fluids are studied from a quasi-molecular point of view with a discrete particle-based model, whereas emulsions from a macroscopic point of view, with a continuum model. It must be highlighted that these systems may be completely interchanged and similar procedures may be used to model polymers as macroscopic systems and emulsions

as microscopic ones. The important aspect here lies in the capability of predicting the fate of the microscopical patterns at different scales and how these patterns influence the entire manufacturing process. Computational Fluid Dynamics (CFD) and Dissipative Particle Dynamics (DPD) were used for the purpose of forecasting and highlight peculiar behaviours of such systems. In this work, concepts regarding these two computational models, strengths and limitations are explained, especially in relation to the complexity of the assessed cases.

1.3 Summary of the work

In this work, we wanted to overcome current difficulties and reluctance against the use of computational model to replace empirical ones. We wanted to prove how it is possible to obtain a good level of prediction for systems composed by structured fluids and emulsions, for which as already mentioned complex interactions take place. Both systems are complicated to assess also from the experimental point of view, and phenomena (such as formation and modification of microphases in structured fluids and droplets breakage and coalescence in emulsions) take place on a small timescale that is usually lost with the level of resolution of most experimental techniques. Also, experiments may be falsified by human errors or misunderstanding. With the support of computational models, the possibility of reproducing a wide variety of different cases and operative conditions gives the chance to explore in more detail the real physics and chemistry behind experiments, but also the capability of improving existing techniques and producing standard protocols of analysis or design of experiments. Still, predictive computational models are not perfect and validation and bench-marking play and important role.

According to the required level of accuracy, different models can be used for simulating completely different systems. In particular, it is possible to identify macro-groups of models such as atomistic/molecular models, coarse-grained/mesoscopic models and continuum models. The first group can be used to reproduce a system at its microscopic scale (i.e. atoms to 10^{-10} m as a length reference and 10^{-8} s as a time reference. The second group can be used to reproduce aggregates of molecules, clusters and micelles (i.e. 10^{-6} as a length ref. and 10^{-4} as a time ref.). The last group can be used to reproduce macroscopic systems (i.e. 10^{-4} m up to km as a length ref. and 10^{-4} up to years as a time ref.). In this work, we focused our attention on two particular models: Computational Fluid Dynamics (CFD) for continuum modeling and Dissipative Particle Dynamics (DPD) for mesoscale modelling. We did not dig into molecular models since we were interested in capturing the evolution of micellar aggregates, that happens at timescales that cannot be easily assessed by microscopic techniques.

Coarse-Grained models, such as DPD, is used in understanding the nature of the microphases that are generated when different compounds are mixed, but also their fate when the fluid is stirred in a mixer or flow in a pipe. However, this technique is

quite recent and yet more validation work is necessary, in order to better understand its limitations. In this thesis, we assessed different aspects and capabilities of this technique, in order to highlight how information can be extracted from this model and used to improve more empirical macroscale models. On the other side, we highlight how equipment can be simulated by using CFD in complex geometries and results can be compared to experimental evidences with a decent level of agreement between experiments and simulations.

To sum up, in this work two different systems are assessed by using two in-silico techniques in order to identify peculiar changes that happen at a mesoscale and are reflected on their macroscopic behavior. DPD is used to reproduce copolymer systems used as surfactant to understand the fate of microaggregates when shear rate is applied on a system, as it happens in industrial mixers. Starting from this point, when the mixture is validated against the experimental results, it is possible to derive important information such as trends in the rheological curves for non-Newtonian fluids, relaxation times, cluster mass distribution and coalescence/breakage rates. In this part of the work, we also introduced an in-house developed open-source clustering algorithm to identify and track the evolution of the aggregates according to the explored simulation setups. CFD is used to reproduce emulsions by using continuum models and that can potentially take information directly from the meso-/microscale. The two techniques and the two systems can be interchanged, meaning that DPD can be used to simulate emulsions and CFD to simulate surfactant. Moreover, as a proof of concept, we have developed an open-source C++ OpenFoam solver that can be used to perform coupled simulations of the two scales. In this code, information is obtained at mesolevel and transferred to a macroscopic level. This final part is still in a validation phase.

1.4 Reading Guidelines

The reminder of this thesis is structured as follows:

- Chapter 1 introduces the aim of the thesis, some fundamental concepts the principal concepts and the industrial context
- Chapter 2 describes different types of soft matter, their applications and manufacturing processes and particular attention is given to the systems investigated in this work. Structured fluids, copolymers in solution and emulsions are discussed with a short description of the most important industrial aspects and rheological problems.
- Chapter 3 contains the mathematical details and the description of the state-of-the-art computational models used in computer simulations. Particular attention is given to Molecular Dynamics, that lies behind the Coarse-grained technique employed in this work, Dissipative Particle Dynamics, and Computational Fluid Dynamics, coupled with Population Balance Models.

- Chapter 4 introduces the solution algorithms of the models introduced in Chapter 3 and the associated computational details.
- Chapter 5 describes the operating conditions employed in simulations. For non-commercial codes, the modified version of the tool, and simulation setups can be easily downloaded from Github, through a QR code. Also, a guide to run the cases is provided.
- Chapter 6 contains all the findings of this work, organized in two sections for the two different scales. In the DPD section, equilibrium results are presented before non-equilibrium results, while in the CFD part, equipment is validated against experimental evidences.
- Chapter 7 contains the discussion regarding the findings, and it highlights how these results can be exploited for further works, but it also shows current limitations of the investigated techniques.

The Thesis is concluded with the listing of routines and codes, that have been developed and used in this work contained in Appendix A. Appendix B contains instead material regarding the solution of the population balance models by using the quadrature method of moment.

Chapter 2

Flowing soft Matter: structured fluids and emulsions

2.1 Introduction

In everyday life, we all are able to recognize and clearly chategorize substances according to the three fundamental states of the matter (i.e. solid, liquid and gas). However, this extremely simple concepts and our capability of distinguish between them are actually quite limited when we move to the products that are used in many fields of application of chemical engineering. For some of them, this distinction becomes less clear, and without exploring extreme conditions (superhot/cold), it is possible to identify intermediate states. If we stick to the common definition of ”fluid”, which implies that even an arbitrary small stress allows flowing effects, familiar substances, such as mayonnaise and window glass are respectively a solid and a liquid. This is true because mayo does not simply flow under the effect of gravity and window glasses may flow (i.e. creeping and ageing effects) because of gravity. This classification may seem counter intuitive, and in order to avoid such confusion, in a field of application where matter can be composed of so many different components, the term ”complex fluid” or ”soft matter” was created. Soft matter refers to all those thick, rubbery, gel and similar substances that cannot be simply classified as purely solid or liquid. According to Larson, 1999, soft matter can be identified as every substance flowing when smooth and modest (at humanly accessible time scale, not in geological ages!) deformation is applied. It can be considered as an intermediate or more a complete definition for such condensed matter that behaves in between solid and liquid states. If we follow this definition, detergents, shampoos, suspensions of colloidal particles, mixtures of polymers, food products, cosmetics and similar fall in this category. In this work, particular attention will be paid to structured fluids made of block-copolymers in solution, i.e. surfactant solutions, emulsions and foams.

One of the most important aspects of soft matter is represented by the response to shear resulting in flowing effects, which translates into complex rheology. Their

rheological response may drive the use of one of these fluids for some applications or others, as it happens in cosmetic, home and personal care, and food industries. In such fields, the final viscosity may determine the failure of one product compared to its competitors. Not only the final viscosity is important but also the rheological behavior during the manufacturing process along the different parts of the processing line is crucial. These fluids may exhibit changes in the viscosity which have to be considered in order to tune their flowing in the process, control mixing and save energy.

In this work, two main systems have been assessed, structured fluids and emulsions. Here, basic concepts related to soft matter are introduced and a specific section is provided for each of these system. In particular, these sections are about the typology of microphases that can be identified, the rheology and the industrial applications.

2.1.1 Structured fluids, copolymers: self-Assembly and microstructures

Structured fluids can be obtained by mixing species with different properties. According to their compatibility with one componet or another, the variety of microstructures that can be obtained is elevated. The term microstructure refers to the shape and dimension of the molecular aggregates or molecular clusters that form in a structured fluid (Jain and Bates, 2003a; Larson, 1999; M., 2013; Sollich et al., 1997). In particular, if one looks at the case of amphiphilic co-polymers with surfactant character (i.e. compounds that can lower the interfacial tension in a mixture; they are composed of an hydrophobic part and an hydrophilic part) in water, it is possible to observe how molecules organize their micelles in the mixture, such that peculiar structures can be identified at different concentrations. An example is given by copolymer micelles in water. Micelles are microstructures that are formed by a core (hydrophobic part of the surfactant) surrounded by a shell (hydrophilic part). Concentration, temperature and pH are just some of the parameters that influence the formation of these aggregates. Also, these structures or aggregates may be different because of their shape, dimension and orientation. They can be defined as microphases, and each of these is related to macroscopic properties of a mixture. The different kinds of microstructure will be assessed in each part of this chapter in relation to the reference system. From an experimental point of view many techniques may be used as an aid to provide a complete characterization of the system. These techniques are: microscopy, SANS (small-angle neutron scattering), SAXS (small angle x-ray scattering), and polarimetry. Optical microscopy allowed scientists to see structures up to 0.5˜ μm, but if electron (scanning electron microscopy or transmission electron microscopy) microscopy is used, it is possible to observe distances up to 1.5̃nm. SANS and SAXS are the most common used experimental techniques. These techniques consist in the evaluation of the change in the direction of the radiation when it passes through complex reticula. By using these techniques, it is possible to appreciate up to 1̃nm of length. Larger structures, between 0.1̃ and 100˜ μ m, can be instead observed by using dynamic light

scattering, while for smaller strctures it is necessary to use x-ray. Finally, polarimetry exploits the capability of oriented microstructures to rotate the plane of oscillation of polarized light. It is important to highlight that if the amount together with the dimension of the structures is extremely high, specific techniques, such as Raman spectroscopy can be used to identify the composition of each structure. The outcome of many experimental techniques can be the reproduction of phase diagrams. Thanks to these diagrams, it is possible to identify microphases that are generated when the spectrum of concentrations of one component into another is explored. Also, peculiar rheological behavior can be extracted from these diagrams.

2.1.2 Rheology

Rheology and viscosity are indicators of how matter flows and behaves under shear (Malkin and Isayev, 2011; Tadros, 2010; Wang, 2017). In simple terms, they are a way to express how soft or hard matter is. There are many ways to retrieve the viscosity of soft matter, mostly grouped in two categories: drag driven and pressure driven methods. Among the first class of methods, sliding plates (planar Couette), concentric cylinders (Taylor - Couette flow), cone/plate and the slit (plane Poiseuille) are the most important. In all these geometries, the induced flow is viscometric, i.e. a constant, or almost constant shear stress is exerted on every small element of the sample.

The velocity imposed on each sample together with the geometry play a fundamental role in defining the value of the shear acting on the system. If we look, for example, at the case of the planar moving plates, the shear rate can be obtained by dividing the velocity imposed on the moving walls by the distance among the two walls. A shear viscosity can be derived by dividing the shear stress (force of a fluid on a surface per unit area in the direction of the flow) by the shear rate.

When the applied shear is constant and applied for long time, the system reaches a sort of steady state, even if in dynamic conditions. However, when it comes to soft matter, steady shear viscosity is not enough to characterize a system. When any kind of stress is applied on the system, at the beginning of the event but also during the whole test, the viscosity may change during time. This is the case of start-up flows. In fact, it is possible to record the evolution of the viscosity in time, since the applied shear is constant but the recorded shear stress of the fluid is time dependent. The viscosity obtained in this way is called transient shear viscosity. One example of transient shear viscosity can be provided by creep tests for glassy or plastic materials, where the temperature of the system is increased to exaggerate the flowing response of the system because of this variation in the viscosity during time.

If one wants to test the response of a structured fluid or an emulsion without modifying the microstructures (i.e. the system is not significantly deformed), it is possible to apply small-amplitude oscillatory shearing. These experiments can be performed in plate-cone geometries. Sinusoidal oscillations of the cone around the axis generate a shear rate that itself is a sinusoidal function in time. Within such

experimental setup, the microstructures present in the fluid can freely arrange themselves (equilibrium structures reached after the *relaxation* time has elapsed), without the formation of anomalous structures induced by strong deformations.

The viscosity may increase (shear-thickening) or decrease (shear-thinning) when the value of the imposed shear increases. When the viscosity of the system is not affected by the applied shear, the fluid is Newtonian. Structured fluids and emulsions behave as intermediate between pure liquids and solids and exhibit both viscous and elastic behaviors, moreover the viscosity can be either Newtonian or non-Newtonian, if one explores different regimes of flow or concentration of components (Gentile et al., 2014; Newby et al., 2009. These systems often exhibit a specific pattern when the viscosity is reported against the applied shear rate. It drops quickly when the value of the shear is high, but, in the low shear region, a plateau is usually identified. The viscosity obtained in this area is called *zero shear viscosity*, η_0. Zero shear viscosity is an important value because it can be related to the relaxation time (longest time for the elastic structures to relax) of different systems. A rule of thumb for systems composed by spherical aggregates reports:

$$\eta_0 = nk_bT, \tag{2.1}$$

where the zero-shear viscosity is obtained by the number of structures recognized per unit volume times the Boltzmann constant and the temperature. The shear viscosity has usually the same value of the complex viscosity obtained from oscillatory experiments. This rule is called Cox-Merz and is valid for many systems, but when it comes to polymers, this rule is not valid. The necessity to discern between the relaxation time and a characteristic time of the flow was translated into the *Deborah* number:

$$\mathrm{De} = \frac{\tau_r}{\tau_f}, \tag{2.2}$$

where τ_r is the relaxation time of the structure and τ_f is a characteristic time of the fluid. When this ratio is much lower than one, the fluid can be considered Newtonian.Elastic effects become more and more relevant as this ratio is higher. Also *Wi*, the *Weissenberg* number was defined to highlight some strange effects of polymers when its value was around one:

$$\mathrm{Wi} = \dot{\gamma}\tau_f, \tag{2.3}$$

where $\dot{\gamma}$ is the shear stress.

Shear viscosity is not the only viscosity that can be calculated for complex fluids, because the response to elongation stresses, in the case of fluids composed by many long chains of polymers, is different if the reference axis is changed. It is possible to obtain the elongation viscosity when the two extremities of the test are stretched with

an exponentially increasing velocity, such that the deformation is also exponential and the obtained shear rate, the ratio between these two parameters is constant.

Rheology is a fundamental tool to understand the overall behavior of complex fluids (Jain and Bates, 2003b; Jansen et al., 2001; Moulik and Paul, 1998; Rehage and Hoffmann, 1988) such that manufacturing process and macroscopic properties can be tuned to obtain desired viscosity.

2.1.3 Applications

Structured fluids and emulsions have mechanical properties that derive from both the liquid and solid the states. If we consider the capability of a fluid of taking the shape of the body in which is contained, soft matter, in contrast, can keep its original shape for a while and then eventually fill the container. This transition from solid to liquid (even few seconds, minutes up to hours years) places them in the category of viscoelastic fluids. Another property regards the distinction between anisotropic, typical of solids and isotropic, typical of liquids, responses. These two states of matter react differently when a mechanical deformation is applied. If the variation of a mechanical properties depends on the orientation of the perturbation, the system is anisotropic, if not, the system is isotropic. For example, liquid crystals, also classified as soft matter, are able to flow as liquids but they have anisotropic properties as solids. Another example is that of surfactant copolymers in solution that are viscoelastic and anisotropic. A last example is an emulsion, notably moyonnaise, an emulsion of oil in vinegar stabilized by lechitin contained in egg yolk that can hold its shape under gravitational effects. In the food industry, many other examples can be recognized such as ice cream, (a mixture of sugar, cream and milk), mustard, cheese, derived from milk which is already a complex fluid containing micelles, gelatin, gel obtained from collagen and water, and chocolate.

Another important field of application regards home and personal care industries. Starting from toothpaste, that can be classified into the soft matter category due to its capability of flowing under modest shear, up to deodorants, shampoos, detergents, conditioners, and body/skin creams. Shampoos can flow from their bottles but they are ”strong” enough not to drip when applied on our bodies and form foams when they come in contact with water.

In the polymer industries, numerous examples of soft matter can be found: from the common solid types of plastic, used in packaging, tires, clothes and so on, that can be extruded, manufactured in fibers or melted. Two common examples of polymers are represented by the triblock copolymer Pluronics, that have been used for many years as surfactant in medicine, food and cosmetic industries and polyurethane foams, that find a great number of applications in coating, insulating and as fillers in car bumpers, seats and refrigerators. Waxy crude oils and pulp fiber in paper industry are again examples of soft matters. In the reminder of the chapter we will discuss more in details the two examples of flowing soft matter that are studied in this work: structured fluids and emulsions. (Larson, 1999).

2.2 Structured fluids

If one is familiar with the concept of polymer, chains of unimers (i.e. single repeating units) bonded together, the concept of copolymer is quite straightforward (Carraher, 2017; Guo, 2016). A copolymer is composed by different unimers that are linked together according to a specific pattern. For example, the most common example of copolymer is formed by alternate blocks of A-type and B-type unimers. The length of the chain determines the kind of complex fluid and it can be ideally endless. Many unimers can form copolymers and two or more of them can be the same, as it happens in the case of the triblock copolymers $A_nB_mA_n$, formed by repeating blocks of A-type linked to B-type and again to A-type. Branched architectures are also possible, such that the variety of copolymers that can be synthesized is incredibly wide. By varying the length of the chains and the unimers present in each chain, one is allowed to confer specific properties to the resulting fluid obtained by dissolving the polymer in a solvent (e.g. water) or by considering the polymer melt. An important feature that is peculiar of copolymers is their capability to microseparate from each other aggregating together forming super-molecular structures or clusters resulting in a wide range of patterns and configurations. Each of these pattern or phase has peculiar properties that are generally linked to a specific product or process. This means that the equipment and the processing conditions are tuned such that only specific phases, orientations or states of aggregation are obtained at the end of the process. The process related to microphase creation and modification is called *self-assembly* (Lavino et al., 2015; Pelesko, 2007; Sundararajan, 2016). Thanks to the high variety of structures, it is possible to obtain network-like, gels, rigid and tough materials that can be exploited for different applications. One use of copolymer is related to their capability of acting as surfactants, by lowering the interfacial tension of two immiscible components. This is the case of the triblock copolymer Pluronics constituted by two blocks of polyethylene oxide (PEO) chained to a central block of polypropylene oxide (PPO). When in water, Pluronics can self assemble in different ways, trying to expose the aqueous environment the hydrophilic PEO chains. A fundamental parameter that has been introduced in the study of copolymer is Flory parameter, χ. This parameter measures the solubility or compatibility of a single block in a solvent and tells how the polymer molecules will self-assemble or microseparate. When this parameter is positive, unimers of one specific type have a strong tendency to segregate from the other blocks. χ varies with temperature and in Flory-Huggins theory (Flory, 1941; Huggins, 1942), the product $\chi k_b T$ repesents the interaction energy when A blocks are mixed with B blocks.

2.2.1 Self-assembly and microstructures

With the word microphases, we refer to a set of different configurations of aggregates that can be generated when copolymers are dissolved into a solvent or

when they are in a melt. Due to the different affinity between the blocks of the polymer and the surrounding solvent, the variety of observed patterns at the microscopic level is huge. Based on the level of aggregation and orientation the structures, it is possible to recognize spherical micellar structures, cylindrical aggregates and lamellae (i.e. alternate sheets). In each of these phases, that can be obtained by varying the volume fraction of each component, the super molecular structures or clusters formed can arrange themselves in peculiar patterns. For example, if one looks at a mixture of water and triblock copolymer, where the triblock is amphiphilic, in the spherical micellar region, hydrophobic groups are shielded against the water environment by hydrophilic groups, and that form a shell around the hydrophobic core. In the hexagonal region (that can be found at higher concentration), the behavior is similar, but single spherical aggregates merge into more complex yet ordinate structures (i.e. cylindrical, rod-like micelles), oriented into an hexagonal matrix, where each cylinder is the vertex of an hexagon. At even higher concentrations, lamellae can be obtained. Lamellar phase is obtained when sheets of copolymer alternate to sheets of water. In this specific chase, each chain of copolymer is stretched such that the hydrophobic tails of one chain are in contact with two (top and bottom) layers of water. However, these structures are not enough to cover the whole range of microphases that can be recognized moving along different concentrations of a mixture of copolymer into a solvent. In fact, the transition between spherical and hexagons is not instantaneous but spherical aggregates may coalesce into worm-like, elongated structures that lose their sphericity and ordered pattern. Also hexagons can merge into intermediate pierced sheets before moving to lamellae. Finally, vesicles are also peculiar aggregates that can be obtained when elongated cylinders close on themselves ”trapping” a secondary phase inside. Ordering is also an important aspect that varies with the concentration. If one looks at the micellar state, for some copolymer, at low-intermediate concentrations (e.g. around 10%), spherical micells are not randomly oriented but they may position themselves into ordered structures (body centered cubic, face centered cubic). Temperature also plays an important role in the transition between ordered and disordered structures. An example of different phases is reported in the following figure. In particular, four phases can be identified: micelles, cylinders, disordered lamellae and lamellae. These phases are obtained starting from the same components by varying the concentration and keeping the temperature constant. These examples, together with ordered micellar structures and reverse-micelle are the most important in completely define a phase diagram.

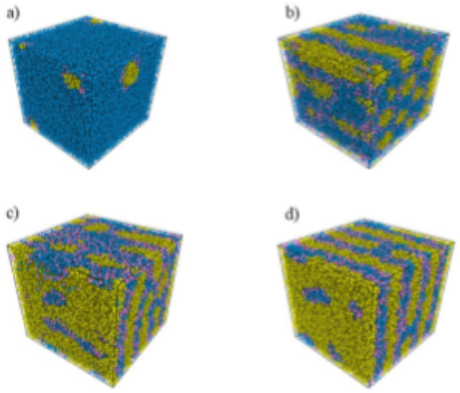


Figure 2.1. *Phases that can be obtained by varying the concentration of a copolymer in water. a) micelles, b) cylinders, c) desordered-lamellae d) ordered lamellae. These phases can exhibit completely different rheological behaviour due to the size and shape of the aggregates that are formed (Pasquino et al., 2019).*

2.2.2 Mixing

Mixing (i.e. the transfer of energy through mechanical agitation) is one of the most important process in manufacturing structured fluids and emulsions. Two or more immiscible fluids can be mixed together by using a specific equipment in order to obtain stable products that must last for a long time without losing their properties. This is the case of shampoos, conditioners, detergents and similar, where their shelf-life has to be granted for several months or even years. For these industries where the complexity of the products is high due to the high number of species that are present in one recipe, it becomes fundamental to achieve perfect mixing, hence produce stable solutions in the smallest possible amount of time. Parameters, such as the dimension of the microstructures that arise during the process become incredibly important in the characterization of one formulation or another, because they can affect macroscopical properties, such as viscosity, mass transfer and stability of a mixture.

Mixing equipment can be different based on the required size of the microstructures, but are mostly stirred tanks (i.e. vessels containing impellers and baffles) and rotor-stator devices. One of the most common problems in calibrating such devices is related to the scale-up from lab to factory scale. It is crucial to identify the relevant set of parameters for each device that can be used as a reference in these scaling operations. For example, the use in-line rotor-stator mixers is quite common, but the physics behind such mixers, is quite unknown, so that the tuning is mostly based on experience (e.g. Hall, Cooke, El-Hamouz, et al., 2011; Hall, Cooke, Pacek, et al., 2011; James, Cooke, A. Kowalski, et al., 2017; A. J. Kowalski et al., 2011).

2.2.3 Rheology

Rheology of structured fluids constituted by copolymers is a non-trivial topic (Tadros, 2010; Wang, 2017). It depends on many different variables such as the type and number of blocks that form the copolymer, the size, shape and orientation of the microphases obtained, the level of interaction between the copolymer and the other media present in the mixing, entanglement and branching of the chains, to cite few of them. A general distinction sees two different behaviours at low shear and high shear. In the extremely low shear limit, the ordered aggregates allow the copolymer to behave as a solid in many cases, while in the high shear regime, copolymers behave as homopolymers and in the specific case of well-aligned lamellae, they behave as a true fluid system. An intermediate behaviour, between pure solid and pure liquid, known as "gel-like", can be observed when hexagons and more generally cylindrical aggregates are obtained. In this specific case, it is possible that the viscosity depends on the kind of stress that is applied. In fact, when the stress is perpendicular to the oriented cylinders, the fluid behaves as a solid, and when the shear is aligned with the direction of the shear, they simply slide and a liquid behaviour can be observed. Ordered structures can be obtained also when a flow field is applied, such that structures are oriented according to the general flow. On the contrary, disordered structures or ordered domains can be observed when equilibrium is reached without any external force acting on the sample. In this case, ordered structures can be appreciated only in small regions (grains).

The capability of changing the viscosity by alignment of copolymer gives the possibility of obtaining structured fluids with high anisotropic behaviours. In the lamellar phase, shearing flow can be parallel, perpendicular and transverse. In the first case, the normal vector of each sheet is perpendicular to the velocity vector and the velocity vector lies on the plane of the sheet. In the second case, the normal of the sheets is parallel to the vorticity axis, whereas in the last case the normal vector of the sheets is parallel to the velocity vector. The three orientations produce variations in the viscosity of the copolymer aggregate.

Defects present in the sheets, due for example to rapid quenching (fast cooling or heating), can produce holes that are also responsible for variation in the viscosity. When cylinders are present, the behavior and the possibility of having different orientations resemble the lamellar phase, but in general the cylinders seem always to be aligned with the flow. In many applications, cylinders can also have different dimension (thicker or thinner) and not just a unique size. A variation in the viscosity can be appreciated also in this domain, when there is a transition between ordered and disordered structures. Finally, in the case of spherical aggregates, when the applied shear is small, fluids behave as pure liquids, and micelles interact according to the hard sphere collision theory. Viscoelastic behavior can be appreciated when the shear becomes extremely high and disordered structures or modification of the structures is achieved.

2.2.4 Applications

The possibility of tuning physical properties, based on the length of the polymeric chains, allows manufacturing an incredible number of products and copolymers can be used in many fields of application in the chemical industry. It is possible to observe how the formation of peculiar microphases is required for example in drugs encapsulation, personal and home care, food industry. An example can be provided by triblock copolymer such as Pluronics mixed with water. At low concentration of Pluronics, spherical micells are obtained. When an organic hydrophobic phase (Lince et al., 2008; Valente et al., 2012) is added to the mixture, this phase is readily trapped into the core of the micelle and these aggregates can be used as micro-carrier for medical applications. In fact, it is possible to drive the velocity of drug release by varying the concentration of copolymer in the mixture. In biotech and especially in fermenters, copolymer membranes can be used to protect microorganisms during the stirring without influencing the growth of the species. Gels can be formed and used because of special rheological properties (mixtures behave as semi-solids) in food and cosmetic industries.

2.2.5 Water/Pluronics Mixtures

In this work, we focused our attention on one specific type of structured fluid, originated from the mixing of water and Pluronics (Alexandridis, 2000, 1997), a class of polymers that behave as surfactant. The term "Pluronics" refers to a class of non-ionic tri-block copolymers, widely investigated (e.g. Aydin et al., 2016; Cao et al., 2005; Cheng et al., 2015; Y. Li et al., 2012; Nicolaides, 2001; Song et al., 2016; Sun et al., 2013), made by already mentioned PEO and PPO, following an A-B-A rule (see Fig.2.2 below):

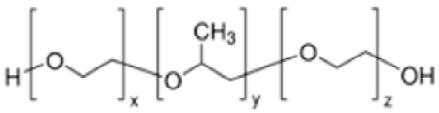


Figure 2.2. *Pluronics structure: tri-block copolymer, A-B-A structure. The two tails are formed by PEO (slightly hydrophilic) repeating groups while the central part is made by PPO repeating groups (slightly hydrophobic).*

Pluronics-type copolymers are used every day as surfactants in many products for the personal care and pharmaceutical industries because of their amphiphilic behaviour. This means that they tend to form microstructures, based on the concentration of copolymer in solvent (e.g. water), such as spherical micelles, cylinders and lamellae. In Fig. 2.3 is reported a grid of the dierent kind of Pluronics based on the percentage of ethylene oxide present in the copolymer.

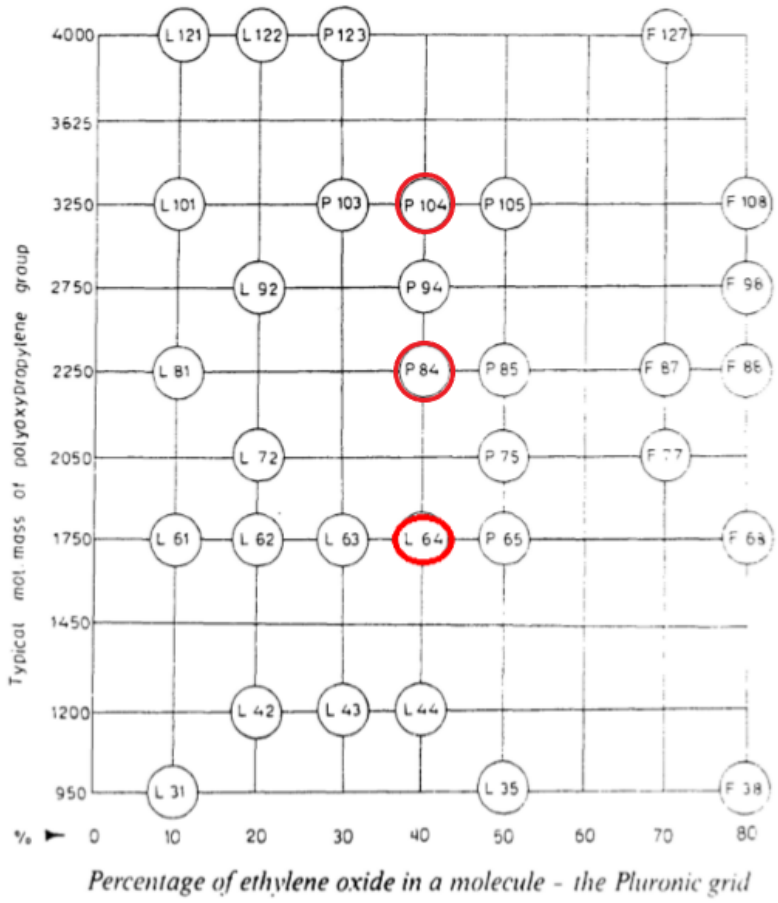


Figure 2.3. *On the x-axis, the increasing percentage of PEO groups into a chain, while on the y-axis the molecular mass of the PPO group in one chain (Alexandridis, 1997). Types of Pluronics classified by concentration of hydrophilic part. Different labels, based on their state of aggregation (P: paste, F: flake, L: liquid), are used. Numbers refer instead to the percentage of the hydrophobic vs hydrophilic chains (i.e. last number is the percentage of PEO divided by ten, the remaining numbers represent the molecular weight divided by three hundred. In the graph, Pluronics L64, P104 and P85 is circled in red.*

For example, Pluronics L64, P104, and P85 are different names that identify three Pluronics, where the phyisical state is Liquid or Paste and the length of the chain, hence the ratio between the hydrophobic and hydrophyilic part, varies. The name of the Pluronics is representative of the physical structure of the surfactant. The letters such as ”L,P,F” mean Liquid, Paste and Flake; the rst number indicates approximatively the weight of the PPO part divided by three hundred; the last number, instead, is the percentage of PEO part divided by ten. For example, L64 indicates a liquid Pluronics, with molecular weight of around 1800 g/mol of PPO that contains 40% of PEO.

Both the monomers show hydrophilic behaviour, but the ramification of PPO

groups, causes a slight hydrophobicity. For example, the cloud point (i.e. the separation from water) ranges from 10° C to 100° C while increasing PEO (more hydrophilic) content as it happens for the dissolution rate, which also increases while increasing the number of PEO groups (Alexandridis, Olsson, et al., 1995; Zhou et al., 1996). Both, PEO and PPO, contain slightly hydrophilic groups, and can form hydrogen bonds with the solvent. At higher temperatures, the hydrophilicity of PPO groups decreases, and aggregation phenomena can occur. When temperature and concentration cross critical values, namely the Critical Micellar Concentration (CMC) and the Critical Micellar Temperature (CMT), aggregates of Pluronics in the form of micelles (i.e. hydrophobic core is surrounded by hydrophilic shell) are formed. This amphiphilic feature gives to Pluronics the possibility of forming complex microstructures, when they are mixed with water and organic components. These microstructures can have shapes and dimensions that directly depends on the kind of copolymer and on the solvent. A broad experimental characterization has been performed on such systems to prove how microstructures are related to experimental conditions, such as concentration, temperature and flowing effects (Alexandridis, Olsson, et al., 1998; Newby et al., 2009; Youssry et al., 2010. In fact, it has been proved that high PEO content or low molecular weight Pluronics do not form these structures at room temperature. The micellization process is mainly driven by entropy and the free energy of micellization is a function of PPO block. When the surfactant is organized in micelles, its $\Delta S_{tens} < 0$. However when is dropped into the solvent (e.g water), whose entropy is much higher, the sum total entropy is greater than zero, $\Delta S_{tens} + \Delta S_{water} \gg 0$ Critical Micelles Concentration (CMC) decreases with increasing PPOs (larger PPO regions show lower CMC) and with increasing temperature, as it is reported in Figure 2.4:

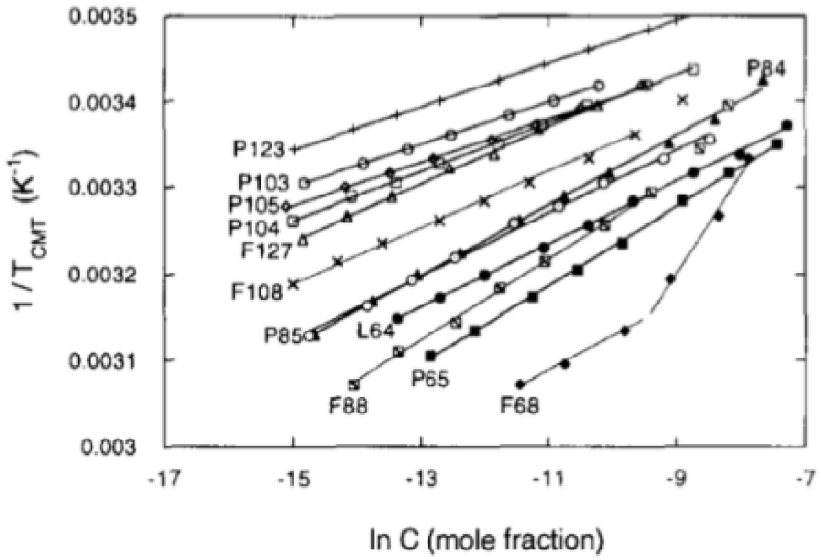


Figure 2.4. *Critical Micellar Temperature for different Pluronics as a function of the natural log of the concentration. In particular, this figures shows how the temperature can affect the formation of micelles (Alexandridis, 1997).*

If Pluronics L64 is considered, the standard enthalpy of micellization ($\Delta\tilde{H}_0$) was found to be around 339 kJ/mol, the standard free energy ($\Delta\tilde{G}_0$) around -28.8 kJ/mol and the standard entropy ($\Delta\tilde{S}_0$) around 1.244 kJ/mol. The fact that the enthalpy of micellization is positive indicates that the process is endothermic. According to experimental studies, L64 was proved to form visible micelles when the concentration of copolymer in water is at least greater than 6%. Micelle size increases with concentration ranging from 10 nm at 8% to 12.5 nm at 20%. The experimental radius of micelles at room temperature ranges from 4 to 5 nm, but it increases when the temperature increases and where the PEO fraction in the polymer chain decreases. At higher temperature and high PPO concentration, the size of the micelle resembles a stretched PPO chain and the shape changes from spherical to rod-like. It has been also reported that at 35℃ the hydrodynamic radius of micelles was almost monodisperse while at lower temperature it was found to be polydisperse.

Monomers can be found at low concentrations and temperatures, meaning that they do not form micelles or complex structures. CMT and CMC, for different Pluronics, depend on the ratio between PPO and PEO groups and, of course, on the solvent. Pluronics that have a ratio of PPO over PEO groups greater than 0.5 form spherical micelles in solution. On the other hand, when this ratio is less than 0.5, aspherical or cylindrical micelles are obtained. As a rule of thumb, Pluronics with smaller amount of PPO tent to form micelles at slightly lower concentrations and temperatures.

Even though they are very common, spherical and aspherical micelles are not

the only microstructures that can be found in the Pluronics phase diagram. While increasing the concentration of Pluronics in water, it is possible to observe a transition from disordered micelles to more complex structures. In their micellar phase, the slightly hydrophobic PPO groups come together, namely the core of the micelle, while the PEO tails orient themselves through the solvent, namely the corona of the micelle.

When a critical value of temperature and concentration is crossed, the number of micelles is high enough to allow PEO groups to create links between them and ordered structures. Typical structures are FCC (face centred cube) or BCC (body centred cube) can be recognized at this stage, which behaves as a solid gel. When the gel is formed, the viscosity of the mixture increases, but small variations of the temperature could destroy these ordered structures and promote the transition from hard- to soft-gel. Cylinders that are already present at intermediate concentrations, can gather and coalesce in order to form bigger and more complex reticulates. If these cylinders arrange themselves into an ordered lattice, the microstructure is named hexagonal (Alexandridis, Olsson, et al., 1995; Holmqvist et al., 1998). At even higher concentrations, the distance between the cylinders becomes so small that they eventually coalesce forming the so called "lamellar phase", where sheets of tri-block copolymer alternate to solvent layers (Almgren et al., 1995; Zhou et al., 1996). Finally, at very high concentrations (close to pure copolymers), a new network is formed, and micelles of solvent are trapped in this structure. Based on the Pluronics type, boundaries between the microphases can be shifted. Also, temperature and flow fields contribute to this shift.

In order to ease the process of labeling all these phases based on the amount of water and polymer, phase diagrams can be experimentally built by using small-angle light scattering (SALS), small-angle X-ray scattering (SAXS), and small-angle neutron scattering (SANS), but also rheological analysis of the response of the system to flow effects.

2.3 Emulsions

As already mentioned, another interesting example of soft matter is emulsion. Emulsions can be defined as an heterogeneous and opaque system composed by (at least) two fluids, defined as disperse and continuous phases (Bernard P. Binks, 1998). The most common example of emulsion is provided by oil-in-water and water-in-oil mixtures. They are present in many applications such as pharmaceutical, oil and gas, detergents, food and cosmetic industries and can be obtained by vigorous agitation, whose energy is able to create droplets of the disperse phase, that are stabilized by addition of surface-active agents (surfactants). The introduction of surfactants in the sysytem is able to minimize the surface tension of the droplets of the disperse phase, and prevents them to coalescence into a bulk phase (Walstra, 1996). Important concepts that are used to better describe the behaviour of the emulsion are the hydrophile-lipophile Balance (HLB), the stability, phase-inversion, formation of gels and so on. These concepts will be presented in what follows.

In order to make an emulsion, three key elements are needed, two immiscible fluids, surfactant and external energy. The combination of these features may produce a rich variety of emulsion with completely different properties, such as the final dimension of the droplets and the final droplet size distribution. A droplet is formed when the difference in the pressure between the inside and the outside crosses a crucial value, defined as Laplace pressure. This critical value is proportional to the surface tension of the two phases and to the inverse of the radii of curvature of the droplet. This is a simple model that highlights the effect of a surfactant, which is able to lower the surface tension, or the deformation needed to form a droplet.

2.3.1 Properties of Emulsions

The concept of HLB has been initially developed, as a rule of thumb, considering only the solubility of the surfactant in one of the two phases. For example, in oil-in-water emulsions that can be obtained by mixing oil, water and a surfactant which is more soluble in water, the HLB was mostly introduced as a way to forecast the type of mixture obtained. For example a system with high HLB forms micelles or similar structures in water (as a continuous phase). This concept becomes more difficult when mixtures are not binary or composed by polymeric chains rather than simple monomers. Also, temperature and different experimental conditions play an important role in the stability and solubility of surfactant into water and oil. An evolution of the concept of HLB, is the HLB related to the whole system, rather than only to the surfactant.Thanks to this concept, HBL was unrelated to the nature and concentration of the surfactant but only to the type of structures formed in one phase or in another (Walstra and Smulders, 1997). Moreover these concepts but also experimental procedures become more and more difficult as we move from a binary to a tertiary mixture. In those cases (for example water-in-third-in-oil, i.e. a mixture

composed by water, oil and a third phase) the variety of structures that may be originated is even wider and the experimental techniques may fail in capturing all of them or their niche behavior. So, from the experimental point of view, phase diagrams are introduced. They are able to predict which one is the continuous and the disperse phase were initially obtained by conductivity studies, where the total conductivity is obtained by summing the conductivity of each phase by the volume fraction of the component (Walstra, 1996, 1993).

Another fundamental concept in emulsions is their stability, that is mainly related to kinetic aspects. In fact, a stable emulsion does not show any changes in time (e.g. seconds, hours, up to years). Phenomena such as sedimentation (due to the gravitational effects on the droplets), flocculation or coalescence may cause instability of an emulsion.

Sedimentation (or creaming) can be seen as droplets that move under the effect of the gravity or centrifugal forces and accumulate in layers (bulk) separated from the continuous phase. During the creaming process, no coalescence is present, and agitation may bring the system back to its original state. Creaming depends directly on the difference in the density between the phases, the dimension of the droplets and the viscosity of the continuous phase. This means that it is possible to inhibit this phenomenon by producing small droplets or reducing the difference in density between phases.

Flocculation is related to the capacity of droplets of coming close together without destroying the interface between them. Larger droplets obtained in this way could also cause faster creaming, because of their size, and in some cases, the formation of gels. Also, if the droplet size distribution is wide, this can help flocculation because smaller droplets can come closer to the bigger ones much easier.

Coalescence instead involves the fusion between two droplets into a new, unique droplet. In order to obtain coalescence, the layer of continuous phase between two approaching droplets must be such small that it can deform and eventually break to allow the droplets to come in contact. The properties of the thin layer between droplets have been studied in order to obtain parameters that can be related to the macroscopical properties of the emulsion, such as shear and bulk viscosity, surfactant diffusion, surface tension and so on. If no electrostatic forces are present, the rate at which droplets come in contact becomes the driving factor that contributes to coalescence.

Finally, gel emulsions can be created when the volume fraction of the disperse phase is around 0.5 and 0.8. These emulsions are known as biliquid foams, meaning that the spherical aggregates cannot arrange themselves in ordered reticula but they coalesce in more complex structures, i.e. cylinders, lamellae, disordered aggregates or combinations of them that can be trivial to describe. Gels can be experimentally observed by using SAXS, SANS and similar.

2.3.2 Rheology

The simplest example of viscosity of an emulsion can be obtained under the assumption that droplets do not deform. The Taylor's formula reads as follow:

$$\mu = 1 + \frac{1 + \frac{5}{2}\frac{\mu_c}{\mu_d}}{1 + \frac{\mu_c}{\mu_d}\phi} \tag{2.4}$$

where μ_c and μ_d are the viscosity of the pure continuous and disperse phases, and ϕ is the volume fraction of the disperse phase. The ratio between the viscosities can be extremely different if we consider droplets or bubbles. Droplet size distribution and presence of surfactant can influence the overall viscosity of the system. Models have been obtained to reproduce the behavior of small and large droplets but also monodisperse or polydisperse systems.

Flocs can be related to the viscosity of the emulsions. Many studies (e.g. B. P. Binks, 2002; Jain and Bates, 2003b; Janssen and Meijer, 1993) have proved that when the concentration of surfactant is below a certain value, emulsions behave as Newtonian fluids, but if the concentration is higher, flocs may break up, causing non-Newtonian, shear-thinning behaviour. However, when the viscosity of the disperse phase is high enough, these droplets do not directly break into smaller entities, but they are instead deformed into cylindrical shapes and may drop into an high number of smaller daughters, because the viscosity is able to reduce the instability of the interfacial forces.

2.3.3 Manufacturing Emulsions

Emulsions can be manufactured by stirring the different species in vessels by using particular geometries of the impellers that can enhance the mixing by causing the formation of regions where shear stresses are extremely high. Agitation becomes the main driver in providing the energy required to deform each droplet and causing breakage. If one is familiar with the concept of turbulence, in turbulent regime, large eddies can interact with the droplets, by shear stresses. But if the size of the eddies is comparable with the droplets, the rupture is caused by inertial forces. In cases of high viscosity or small geometries, the flow regime becomes laminar, and such that only viscous rupture is present. In this work, the turbulent regime is explored. The presence of eddies means that local fluctuations in the velocity field are present. Large eddies transfer energy to smaller eddies, that in contrast have higher velocity gradients. The smallest size of the eddies that are to transfer energy is called Kolmogorov length scale, and it depends on the viscosity of the continuous phase and on the turbulent dissipation rate, which is produced by agitation. Inertial and viscous forces are responsible for the breakup of the droplets. Based on the dimension and structures obtained at the end of the stirring process, mixtures may be distinguished.

In fact, mixtures of miscible or immiscible fluids can produce a rich variety of structures, that can be appreciated only at a microscopical level. Those structures are responsible for peculiar properties of the mixtures, and they may evolve during the manufacturing process, based on experimental setup, in completely different structures. Forecasting their behavior is of fundamental importance in the research area for such industries, and this is where phase diagrams, previously introduced, come to play.

Chapter 3

Computational Models

3.1 Introduction

In these last years, the concept of lean industry is driving the change and lot of effort is put into reducing wastes, standardizing and on optimizing industrial processes. One important tool, which is supporting this revolution, is the capability of providing computer simulations of industrial processes but also new formulations to better understand the intrinsically present physics and chemistry in all the production lines of the chemical industry, within short time and appropriate accuracy. Simulations, that were invented just as a tool to prove the computational power of the first electrical computers during World War II, allowed operators to easily recover the basics behind experimentally-driven choices but also to improve and tune operating parameters such to reduce human errors. More and more complicated operations were needed to test those machines, hence, new models describing the physics and chemistry in an accurate way, were coded, implemented and tested.

Computer simulations became soon a turning point, because until then, even describing the interactions between more than two bodies using paper and pencil was extremely hard and time consuming. By passing the time, the increased computational power and the development of accessible, cross-platform, coding languages, allowed engineers to unleash their capability to predict and improve real industrial processes. Nowadays, computational engineering is a widespread field of application in many areas (from chemistry, to mechanics, to aerospace, to finance and economy). However, the complexity of some phenomena needs specific tools and techniques in order to obtain exact predictions but also the capability of capturing niche events (e.g. microscopical phenomena, multiphase fluids, complex interactions, ...). This is the natural playground for a variety of models to come to life.

Soon, computer in-silico experiments became a fundamental tool to revise, prove or improve old theories derived only from empirical observations. This new way of coupling experiments to simulations moved science to a deeper level of knowledge and gave the possibility to discover underlying, unexplored events until that point.

Different mathematical models can be used to assess the different scales of the system. The word *scale* here wants to define both a geometrical and temporal scale. If one wants to understand the chemistry behind each process and have a deep understanding of molecular interactions, atomistic or semi-atomistic models are required. Examples of these models are provided by Quantum Chemistry (QC), Molecular Dynamics (MD), Coarse-Grained (CG), and Monte Carlo Simulations (MC). These models consider the true atomic nature of the matter and try to describe their behaviour, via modelling the interactions between atoms or by using statistics.

In this work, the concepts of MD and CG will be described in detail, with particular attention to the connection between the two models. However, by using the above-mentioned models, it is impossible to simulate industrial equipment that is relevant for the chemical industries, where further approximation is needed in order to obtain reasonable results in an acceptable amount of time. For such purpose, Computational Fluid Dynamics (CFD), can be used instead. CFD allowed chemical engineers to characterize flow fields, assess transport phenomena, and optimize the production at industrial scale, without the degree of accuracy of atomistic and semi-atomistic models. The reduced computational time coupled with the possibility of exploring bigger scales and the reliability of the results, however, depend on the accuracy of the approximated models (mostly empirical) that are used. One example can be provided by the calculation of transport coefficients for multiphase systems, simulated at quasi molecular scales and transferred into more complex models, i.e. observing the response of a system when strong perturbations are affecting it.

This latter approach started a new way of thinking about computer simulations. Each scale, with its own level of resolution, can provide partial of complete information on the overall behaviour of a complex system (from the equipment to the mixtures). In this framework, a multiscale modelling approach was developed. The idea behind multiscale modelling is to create, first off-line and then on-line, connections between detached levels of resolution. Information which can be obtained at atomistic/semi-atomistic (MD and CG) level of resolution can be passed to less accurate scales (CFD) to enhance or replace empirical approximation with new concepts derived from first principles. However, the discrepancy between the time and space scales can cause problems in the physical coupling between the scales. This is a critical point of multiscale modelling, that cannot be easily overcome, also because of a practical limitation due to the computational power. However, a need for a completely defined multiscale framework is now clear. In particular we started our analysis by using a semi-atomistic level of resolution and moved to an industrial scale description. In order to have a better understanding on the state of the art of the simulation tools, with advantages and limitations, some techniques are here discussed in detail.

3.2 Full-Atoms and Coarse-Grained Models

In this first part, an overview of the state-of-the-art computational models that are used to describe a system with atomistic/semi-atomistic level of resolution is provided. Particular attention is given to Molecular Dynamics (MD), used as a starting point to derive Coarse-Grained (CG) models, with specific interests to Dissipative Particle Dynamics (DPD). The different models refer to the spatial and time resolution that can be assessed by varying the initial assumptions. In the microscale and with a discrete description of the systems, Hamilton's equations are solved, while in the macroscale where continua are assessed, variables are mostly related to thermodynamics. All the scales in between can be defined as mesoscales, and each of them carries a different level of information.

The smallest scales can be addressed by MD or even QC, if needed. As an example, MD can be used to study complex problems involving proteins, membranes, and polymers, however some phenomena can be only appreciated at timescales that are not commonly accessible by this method (Allen and Tildesley, 2017). In the area of modelling and simulations, this became soon a problem and new concepts and tools to investigate these unknown scales, even with reduced accuracy. Coarse-Grained (CG) models were initially developed to study mostly detergents and membranes, system composed by many different components, interacting between them and forming complex microstructures. The creation, modification or orientation of such structures can cause a strong variation in the transport coefficients (Alexandridis, Olsson, et al., 1998; Gentile et al., 2014; Newby et al., 2009; Youssry et al., 2010). However, all these phenomena happen on a scale which is much larger than the actual capability of MD, but smaller compared to the approximations made in CFD modeling. This becomes exactly the point where CG models can be exploited. CG models can be derived from MD. However, it is important to have in mind some important concepts related to it.

3.2.1 From Liouville to Boltzmann Equation: Mesoscale Modelling

In this part, the governing equations behind micro- and meso-scale modelling are presented. In particular, concepts related to the modelling of discrete elements and their distributions are provided and eventually the well-known *Liouville equation* is derived starting from a microscopic approach (Chiarotti et al., 1990; Gombosi, 1994; Leaf, 1972. This equation is fundamental beacuse it describes how particles interact and it is a bridge from the microscale to the assumption of continua. Before we start, it is necessary to introduce the concepts of *configuration space*, i.e. each particle is defined by its position (i.e. x, y, z coordinates) with M degrees of freedom (i.e. independent variables that define a single state of the configuration space) and the *phase-space* (i.e. the collection of all the possible states of the system). Each point of the phase-space can be represented by a general P($\mathbf{x}, \mathbf{v}$), which is one state defined by positions and

velocities (or momenta).

An element of fluid, comprising an incredibly large number of particles (M> 10^{23}), is the domain of investigation. If one wants to completely describe this domain must define for each of the N-particles, the following quantities:

$$\mathbf{r} = (x, y, z); \mathbf{p} = (v_x, v_y, v_z); \mathbf{s} = (s_1, s_2, s_3 ... s_K); \tag{3.1}$$

where $\mathbf{r}$ is the position vector, $\mathbf{p}$ is the momentum vector, and s_i is an internal degree of freedom (e.g. vibrational modes of molecules). For simplicity, the number of internal degrees of freedom is considered as zero, such that to define a state only 6M d.o.f. are necessary. The microstate of a system composed by M particles (i.e. one point in the phase-space diagram) follows:

$$\overline{\Gamma} = (\mathbf{r_1}, \mathbf{r_2}, ... \mathbf{r_M}, \mathbf{p_1}, \mathbf{p_2}, .. \mathbf{p_M}) \tag{3.2}$$

where $\mathbf{r_i}$ is the position of the i -th molecule and $\mathbf{p_i}$ is the momentum of the i -th molecule. Due to its discrete nature, the system of particles can be only described by using the Hamiltonian formalism:

$$\dot{\mathbf{r}}_\mathbf{i} = \frac{\partial \mathscr{H}}{\partial \mathbf{p_i}}; \dot{\mathbf{p}}_\mathbf{i} = \frac{\partial \mathscr{H}}{\partial \mathbf{r_i}} \tag{3.3}$$

The solution of the Hamiltonian at time t and the equations of motion define a trajectory in the phase-space diagram. Because of the deterministic character of the system, two trajectories cannot cross each other.

This equation can be used to describe any single element of fluid and can provide a complete description of the system. However, due to the incredibly high number of particles and trajectories that must be computed, and the impossibility of defining initial conditions for all the particles at time t, this formulation of the system is useless.

Characterizing a system through the definition of the behaviour of each single particle is also not needed because, most of the time, what is important is their average behaviour and the properties derived from it, such as temperature and density. Here it is possible to reduce the complexity of the high number of microstrates into a single macrostrate through the definition of *ensamble*.

It is now important to clarify that $\mathbf{r_i}$ will be used to describe the position of the i-th particle, while $\mathbf{r}$ will be used to define the position of all the particles, such that $\mathbf{r} = (r_1, r_2, ..., r_M)$, and the same applies for $\mathbf{p_i}$ and $\mathbf{p}$. An ensamble can be described using a *distribution function*, $f^{(N)}(\mathbf{r_i}, \mathbf{p_i})$ of the particles within it, defining the following quantities:

$$\int f^{(M)}(\mathbf{r_i}, \mathbf{p_i}) \prod_{i=1}^{M} d^3\mathbf{r_i} d^2\mathbf{p_i} = 1 \tag{3.4}$$

normalized to one, where the value inside the integral is the probability of finding the system in a specific microstate defined by $\{\mathbf{r_i}, \mathbf{p_i}\}$. The probability of finding all the

microstates is equal to one, meaning that all the microstates are explored. The evolution of the distribution function in time and space can be described according to the Liouville equation, which must respect a continuity equation where the properties change in time, are advected because of the flow, and the total flow within a volume is conserved. Three equations of Liouville can be used to describe the evolution of the probability distribution function:

$$\frac{\partial f^{(M)}}{\partial t} + \sum_{i=1}^{M} \left(\dot{\mathbf{r}}_\mathbf{i} \cdot \frac{\partial f^{(M)}}{\partial \mathbf{r}_\mathbf{i}} + \dot{\mathbf{p}}_\mathbf{i} \cdot \frac{\partial f^{(M)}}{\partial \mathbf{p}_\mathbf{i}} \right) = 0, \tag{3.5}$$

The Liouville equation shows that an element of fluid can be followed in space, and its shape may change but the volume is always constant. The resolution of the Liouville theorem is non-trivial, since probability distribution functions depend on $6 - M$ variables. It is possible to reduce the complexity of the system by assuming the following:

$$\mathbf{w}_\mathbf{i} = (\mathbf{r}_\mathbf{i}, \mathbf{p}_\mathbf{i}) \tag{3.6}$$

and the reduced or K-body distribution function given by:

$$f^{(K)}(\mathbf{w_1}, \mathbf{w_2}, .., \mathbf{w_K}) = \frac{M!}{(M-K)!} \int \prod_{i_K+1}^{M} d^6\mathbf{w}_\mathbf{i} f^{(M)}(w_1, ..w_M, t), \tag{3.7}$$

which for a single particle becomes:

$$f^{(1)}(\mathbf{w_1}, t) = M \int \prod_{i=2}^{M} d^6\mathbf{w}_\mathbf{i} f^{(M)}(\mathbf{w_1}, .., \mathbf{w_M}, t) \tag{3.8}$$

representing the number of particles enclosed in a volume centered in (x, y, z) and (v_x, v_y, v_z) and with volume $d^3\mathbf{r}d^3\mathbf{p}$.

At this point, it is important to say that the expected value of any observable quantity $Q(\mathbf{w})$, depending additively on single particles of the phase space is given by:

$$< Q >= \int d^6\mathbf{w_1}..d^6\mathbf{w_M} f^{(M)}(\mathbf{w_1}, \mathbf{w_2}, .., \mathbf{w_M}) \sum_{i=1}^{N} Q_i, \tag{3.9}$$

Particles are identical and this implies that the distribution function is a symmetric function and indices can be swapped without altering our solutions such that:

$$< Q >= \int d^6\mathbf{w_1} Q(\mathbf{w_1}) f^{(1)}(\mathbf{w_1}) \tag{3.10}$$

meaning that knowing a single probability distribution function (PDF) is sufficient to determine a quantity $Q(\mathbf{w})$. The temporal evolution of the reduced probability distribution function can be re-written using Hamiltonian formulation:

$$\frac{\partial f^{(K)}}{\partial t} = \frac{M!}{(M-K)!} \int \prod_{i=K+1}^{N} d^6\mathbf{w_i} \{H, f^{(M)}\}, \tag{3.11}$$

and for one particle:

$$\frac{\partial f^{(1)}}{\partial t} = \{H^{(1)}, f^{(1)}\} + \left(\frac{\partial f^{(1)}}{\partial t}\right)_{coll} \tag{3.12}$$

being the Hamiltonian term given by:

$$\mathscr{H}(\mathbf{r_i}, \mathbf{p_i}, t) = \sum_{i=1}^{M} \frac{\mathbf{p}^2}{2m} + \sum_{i=1}^{M} V(\mathbf{r_i}) + \sum_{i<j} U(\mathbf{r_i} - \mathbf{r_j}), \tag{3.13}$$

where the Hamiltonian is the sum of a kinetic energy, a potential energy and the energy corresponding to an external force. For one particle the Eq.3.12 can be reduced to

$$\frac{df^{(1)}}{dt} = \left(\frac{\partial f^{(1)}}{\partial t}\right)_{coll}, \tag{3.14}$$

where

$$\left(\frac{\partial f^{(1)}}{\partial t}\right)_{coll} = \int d^3\mathbf{r}_2 d^3\mathbf{p}_2 \frac{\partial U(\mathbf{r} - \mathbf{r}_2)}{\partial r} \cdot \frac{\partial f^{(2)}}{\partial \mathbf{p}}, \tag{3.15}$$

The solution of this equation requires knowledge of $f^{(3)}$ (BBGKY hierarchy). The simplest of the equations belonging to the BBGKY equations is the closed version for $f^{(1)}$, known as the Boltzmann equation:

$$\frac{df^{(1)}}{dt} = \mathscr{C}[f^{(1)}] \tag{3.16}$$

where $\mathscr{C}$ is the collision operator, under the assumption that the probability distribution function (PDF, i.e. the probability of finding a particle in a specific position $\mathbf{r}$ and with a specific velocity $\mathbf{v}$), of two particles $f^{(2)}(\mathbf{r}, \mathbf{r}, \mathbf{p_1}, \mathbf{p_2}) = f^{(1)}(\mathbf{r}, \mathbf{p_1}) f^{(1)}(\mathbf{r}, \mathbf{p_2})$ is given by the product of the PDF of two single particles (i.e. collisions at position $\mathbf{r}$ happen between particles with uncorrelated velocities, under the assumption of the theory of molecular chaos (Bird et al., 2002)).

3.2.2 Molecular Dynamics

When it comes to evaluate the interactions within many bodies, MD becomes a fundamental computational tool (Frenkel and Smit, 1996). In such a context, particles, representing single atoms, can interact via classical mechanics. It becomes possible to simulate a wide variety of materials by neglecting quantum effects. MD spectrum of

applications is large and it ranges from biology, where complex proteins are simulated, to chemistry and to medicine. However, for extremely complex systems, the timescale for relevant phenomena to be observed cannot be assessed by this technique and less accurate ones are instead used (coarse-grained methods). MD tries to fill the gap between information contained at microscale and the empirical data obtained at lab scale. We can have accurate results regarding bulk properties (transport coefficients, rheological properties, spectra) based on our computational time, power and, of course, budget. The main concept behind MD, is to reproduce at a microscopic level, what experiments do at laboratory scale. Different sets of samples are produced and connected to measuring devices, noise is filtered through averaging procedures, and desired behaviours can be detected as a final result. The aim is to reduce the fitting and guessing that is done at lab scale, by assessing directly measuring quantities related to microscopic behavior of such complex systems. In MD samples, atoms can interact via different potentials and moved according to the Newton's equations of motions (Allen and Tildesley, 2017):

$$m\mathbf{a_i} = \mathbf{F_{MD,i}}(\mathbf{r}) \tag{3.17}$$

$$\mathbf{F_{MD,i}}(\mathbf{r}) = -\frac{\mathrm{d}}{\mathrm{d}\mathbf{r_i}}E, \tag{3.18}$$

Integration of the equation of motion, can be performed via different algorithms, and one of the most common is the Velocity-Verlet algorithm (VV):

$$\mathbf{r}(t+\Delta t) = \mathbf{r}(t) + \mathbf{v}(t)\Delta t + \frac{1}{2}\mathbf{a}(t)\Delta t^2 \tag{3.19}$$

$$\mathbf{v}(t+\Delta t) = \mathbf{v}(t) + \frac{\mathbf{a}(t)+\mathbf{a}(t+\Delta t)}{2}\Delta t \tag{3.20}$$

The standard implementation of VV implies that the velocity is updated by half timestep, using the initial velocity and acceleration, then position is fully updated to the next time step using the calculated value of the velocity. Updated acceleration is calculated using the potential, and eventually, the velocity is obtained:

$$\mathbf{v}(t+\frac{1}{2}\Delta t) \rightarrow \mathbf{r}(t+\Delta t) \rightarrow \mathbf{a}(t+\Delta t) \rightarrow \mathbf{v}(t+\Delta t), \tag{3.21}$$

Exact reversibility, low order in time (allowing long timesteps), easy to be implemented in computational codes, one expensive calculation (force evaluated once per timestep), made this algorithm one of the most used in MD simulations and modification to the original version allowed the spreading of new models. Particles representing atoms can interact via bonded and non-bonded interactions. Non-bonded interactions can be obtained by considering one pair of atoms and evaluating the following contribution:

$$E_{nonbonded}(r^N) = \sum_i e(r_i) + \sum_i \sum_{j>i} E(r_{ij}) + .., \tag{3.22}$$

Where $e(r_i)$ can be any external potential field and $E(r_i)$ is the non-bonded interaction potential. As an example, one of the most common type of potential that is used to describe how two non-bonded atoms interact is the Lennard-Jones potential (Lennard-Jones, 1924) and reported in Figure 3.1:

$$E_{LJ}(r_{ij}) = 4\varepsilon_{LJ}\left[\left(\frac{\sigma_{LJ}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{LJ}}{r_{ij}}\right)^{6}\right], \tag{3.23}$$

where $r_{ij} = |\mathbf{r_i} - \mathbf{r_j}|$,

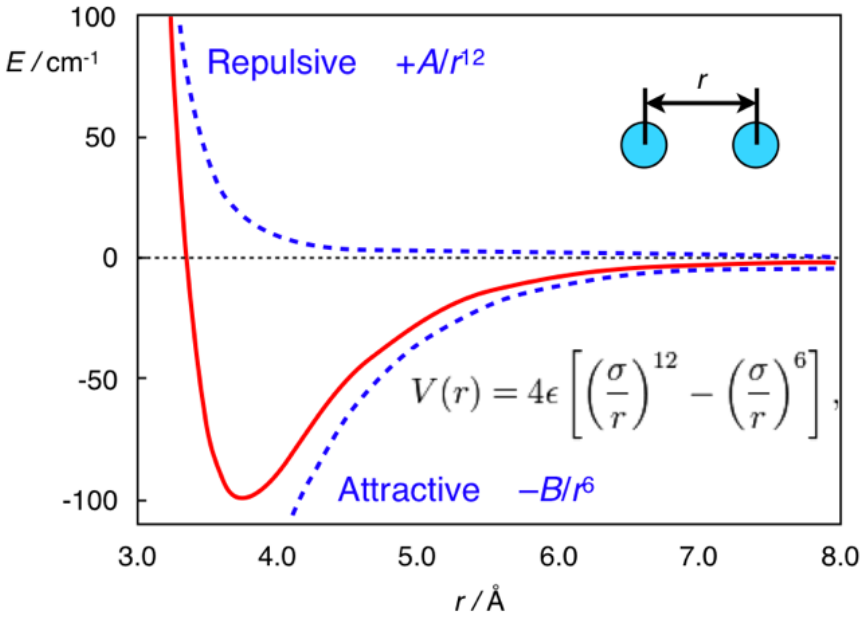


Figure 3.1. *The Lennard-Jones potential.*
source:https://chemistry.stackexchange.com/questions/34214/physical-significance-of-double-well-potential-in-quantum-bonding

Where ε is the depth of the potential well, σ is the finite distance at which the potential is zero, r is the distance between atoms. Molecules can also be simulated using MD. For such purpose, bonded interactions must be taken into account in evaluating the value of the force acting on each particle. In its general formulation, the bonded potential has the following expression:

$$E_{intra} = E_{bonds} + E_{bend} + E_{torsion}, \tag{3.24}$$

As an example, E_{bonds} can be expressed as harmonic potential:

$$E_{bonds} = \frac{1}{2} \sum_{N_{bonds}} K_{ij}(r_{ij} - r_{eq})^2, \tag{3.25}$$

where K_{ij} is the harmonic constant, r_{eq} is the equilibrium distance between two bonded particles.

An important aspect in presenting MD and similar methods, lies in the concept of periodic boundary conditions (see Fig. 3.2). If one tries to simulate complex systems, containing a huge number of exactly identical particles (atoms, molecules), it may be extremely computational expensive to consider all of them. Also, MD simulations are performed in closed boxes containing the different particles, meaning that many atoms will be located at outer faces of the system. These outer layers could negatively affect the calculation of some properties and nullify the validity of the simulation. To avoid this problem, replicas of the original box can replace each edge of the initial simulation region. In such a way, if an atom leaves the box from one side, an identical copy enters the box on the opposite side. Using periodic boundary conditions, atoms can also interact with out-positioned atoms, or better, images of atoms that are a replica of the originals. A neighboring list can be created at every timestep (or updated when necessary) because only pairwise interactions are considered. This means that each atom in the box has a certain number of possible interactions based on a cut-off distance. Only the interactions between the atom i and the j-s included into the list are evaluated. When flow effects are applied on the system, it may be useful to update this neighboring list every timestep, because particles can move apart from each other after every single timestep, hence interactions may be lost.

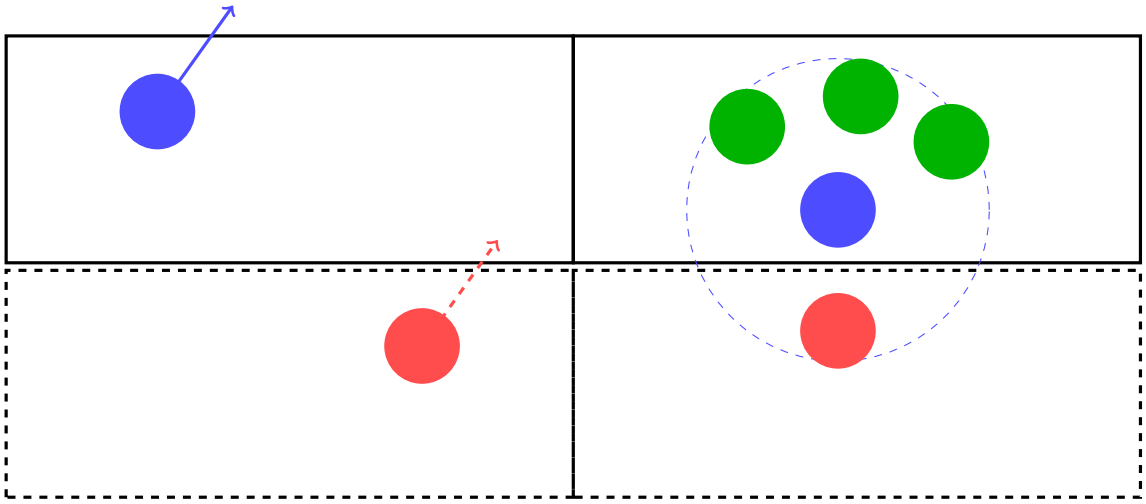


Figure 3.2. *On the left: representation of the periodic boundary condition. When a particle (black) leaves the simulation box (solid lines) from one face/side, a new particle enters in the simulation box from a corresponding position of a replica of the simulation box located on the opposite face/side. On the right: representation of the concept of neighboring list. Each particle (blue) can interact with other particles (green) surrounding it within a certain radius of interaction. When a particle is located on the boundary, it can also interact with replicas of the particle provided by the periodic boundary condition (red), in order to reproduce bulk effects.*

3.2.3 The Langevin Equation

Introducing the Langevin equation is fundamental in order to understand the transition from atomistic to mesoscopic (coarse-grained) models. It is generally used to describe Brownian motion, which was initially described by A. Einstein, who related for the first time the diffusion constant to atomic properties (Langevin, 1908). In order to obtain the Langevin equation, a Brownian particle is immersed in a control volume comprising many smaller particles. If the Brownian particle is bigger than the remaining particles (i.e. they are as smaller as atoms), its agitated motion will be slower compared to the others. In this systems, it is possible to assume that different timescales are present, in particular one related to the motion of the atoms, one to the the Brownian particle to relax its velocity and one to diffuse of a distance at least as big as its radius. As an example, the timescale related to atoms fluctuations is in the order of magnitude of:

$$\tau_s = \text{atoms fluctuations} = 10^{-12}\text{s}; \tag{3.26}$$

the motion of a particle must obey the Newton's law and the force acting on the Brownian particle at time t is due to the interaction between the big particle and the surrounding fluid composed by atoms. This force is however driven by a frictional effect and a stochastic contribution due to random density fluctuations in the fluid such that we can derive a model to describe the Brownian motion of a particle:

$$\frac{d\mathbf{r}(t)}{dt} = \mathbf{v}(t) \tag{3.27}$$

$$\frac{d\mathbf{v}(t)}{dt} = -\frac{\gamma}{m}m(t) + \frac{1}{m}\eta(t) \tag{3.28}$$

where γ is given by $6\pi\mathring{\eta}$ (being η representative of the fluid viscosity) according to the Stokes law. The role of the stochastic force is to kick the Brownian particle, otherwise its velocity would decay to zero within a certain time t. The physical explanation is that atoms in the surrounding fluid randomly hit the Brownian particle keeping the velocity different from zero.

3.2.4 Coarse-Grained Models

In CG models, clusters of atoms are considered as single particles (or beads), and the interactions between pairs of atoms are replaced by interactions between beads. Even if the atomic resolution is lost, the basis of such methods lies in the underlying molecular aspects. In the clustering procedure, the degrees of freedom of the system are reduced. In fact, if a particle (atom or molecule) was able to move freely along x,y, and z coordinates before, now it belongs to a new entity (bead) together with similar atoms and they move only according to the overall bead motion. The clear advantage of this technique is that by coarse-graining we are able to access much larger time span

compared to MD in less computational time. In order to validate results obtained from CG procedures, results are compared against experimental evidences (a-posteriori) (Di Pasquale et al., 2019).

Complex systems may be described by using many degrees of freedom, and their number is directly related to the level of information (or scale) that we can derive from them. By defining these peculiar sets of variables, the state of a system can be completely defined. If one reduces the number of those degrees of freedom (by coarse-graining), the number of these variables can be reduced and some of the information is lost. Dynamic equations are given for each specific variable and different time scales can have different sets. A detailed definition of these specific time scales allows a correct CG procedure, that is able to detect phenomena that can be observed only on an equal or bigger time scale.

The loss of degrees of freedom is reflected into stochastic terms which appear into the equations describing the evolution of the system, simply because it is impossible to go back to the initial stage, since many different coarse-grained initial stages are compatible with a current microscopic state but also it is impossible to predict the future states of such systems. If stochastic terms are introduced in the coarse-graining procedure, the dynamic equation can be derived in the form of Fokker-Planck equation, where together with the stochastic term, a systematic dissipative variable is introduced. This dissipative term, related to the thermal fluctuations, cannot be independent from the stochastic one, therefore there exists a correlation, formulated in the fluctuation dissipation theorem (Kubo, 1957), that links these two variables. The Fokker-Plank equation, which is the base of coarse-grained models, can be derived from Liouville theorem. Starting from a microscopic description, we can define classical equations using Hamiltonian notation:

$$\dot{\mathbf{r}_\mathbf{i}} = \frac{\partial \mathscr{H}(z)}{\partial p_i}; \dot{\mathbf{p}_\mathbf{i}} = \frac{-\partial \mathscr{H}(z)}{\partial r_i}; \dot{z} = L_0 \frac{-\partial \mathscr{H}(z)}{\partial z}; L_0 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \tag{3.29}$$

Where the initial microstate of the system can be completely defined by knowing $\mathbf{z} = (\mathbf{r_i}, \mathbf{p_i})$ and the set of all z defines the phase space of the system. In order to solve Hamiltonian equations, the initial state $\mathbf{z_0}$ must be identified, even if defining this value can be non-trivial. A solution of the Hamilton's equations can be $z(z_0, t) = T_t z_0$, where T_t is a time evolution operator. If we define ρ_t, $Lio(z)$, as the probability distribution function of one state z at time t, and M the region of non-vanishing states, the Liouville equation can be obtained:

$$\int_M \rho_{Lio}(z,0) dz = \int_{T_t M} \rho_{Lio}(z,t) dz, \tag{3.30}$$

By performing a changing in variables, integrating and then taking the derivatives of both terms (omitted for brevity), the solution of the equation is the following:

$$\frac{\mathrm{d}\rho_{Lio}(T_t z,t)}{\mathrm{d}t} = \frac{\mathrm{d}\rho_{Lio}(z,0)}{\mathrm{d}t} = 0 \tag{3.31}$$

And introducing the Liouville operator results in:

$$iL = \sum_i \left(\frac{\partial \mathscr{H}}{\partial p_i}\frac{\partial}{\partial r_i} - \frac{\partial \mathscr{H}}{\partial r_i}\frac{\partial}{\partial p_i} \right) \tag{3.32}$$

Previous equation can be written as follows:

$$\frac{\partial \rho_{Lio}(z,t)}{\partial t} = -iL\rho_{Lio}(z,t) \tag{3.33}$$

The Liouville equations becomes the base to derive the Fokker-Plank (FP) equation. In this work, the complete derivation is not reported, instead the final form of the FP equation for the probability distribution function, which is a general form of the Green-Kubo equation, is reported:

$$\partial_t P(\mathbf{r},t) = -\frac{\partial}{\partial r_i} \cdot \mathbf{v_i}(r) P(r,t) + \frac{\partial}{\partial r}\Omega(r) D_{ij}(r) \cdot \frac{\partial}{\partial \mathbf{r_i}} \frac{P(\mathbf{r},t)}{\Omega(r)} \tag{3.34}$$

More details can be found in Espanol, 2004, but it is important to highlight that thanks to this formulation for coarse-grained models, transport coefficients can be derived.

3.2.5 Dissipative Particle Dynamics

Dissipative Particle Dynamics was firstly introduced to Hoogerbrugge and Koelman (Hoogerbrugge and Koelman, 1992) as an alternative to lattice-gas automata. This model was proposed as a different tool compared to MD that was able to to simulate complex hydrodynamic behaviour of fluids with shorter computational times. During the years, the initial DPD model was corrected and improved by many contributors, and among them, it is important to underline the work of Espanol, Pagonabarraga and Warren (Español, 1995; Groot and Warren, 1997; Soddemann et al., 2003). DPD belongs to a class of methods related to coarse-grained models, in which, compared to MD, the atomistic resolution is lost, and classical mechanics is used to describe the evolution of the particles composing the system. Those aggregates (beads) are not single atoms, as it is in MD, but instead they are representative of clusters of atoms, molecules or parts of molecules, that move together and interact via only-repulsive potentials. All the beads, present into a simulation box, move according to the Liouville equations of motion:

$$\frac{\mathrm{d}\mathbf{r_i}}{\mathrm{d}t} = \mathbf{v_i}, \tag{3.35}$$

$$\frac{\mathrm{d}\mathbf{v_i}}{\mathrm{d}t} = \frac{\mathbf{f_i}}{m_i}, \tag{3.36}$$

where $\mathbf{r_i}$ is the position of the i -th particle, $\mathbf{v_i}$ is its velocity, t is the time and $\mathbf{f_i}$ is the sum of the forces, acting on each bead that can be obtained as follows:

$$\mathbf{f_i} = \sum_{i \not\equiv j} (\mathbf{F_{ij}^C} + \mathbf{F_{ij}^R} + \mathbf{F_{ij}^D}), \tag{3.37}$$

representing respectively conservative, stochastic and dissipative forces. The conservative force can be described as follows:

$$\mathbf{F_{ij}^C} = \begin{cases} a_{ij}(1 - \frac{\mathbf{r_{ij}}}{\mathbf{r_c}})\hat{r_{ij}} & r_{ij} < r_c \\ 0 & \mathbf{r_{ij}} > r_c \end{cases} \tag{3.38}$$

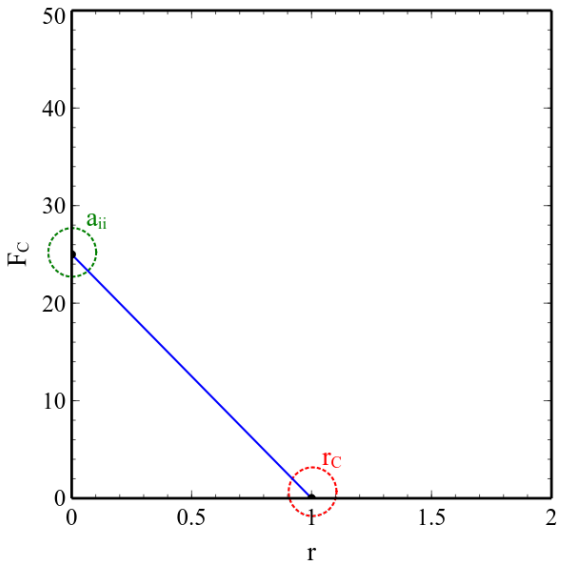


Figure 3.3. *Representation of the force acting between two beads interacting with a soft potential. In this specific case, the maximum value of the force is equal to 25, corresponding to $r_{ij} = 0$, meaning that in contrast with the LJ potential, when the distance between the two particles is zero, the value of the potential is still bounded and the force has a finite value. DPD particles, due to the form of the potential, can indeed cross each other or overlap. When two beads move apart from each other such that the distance between them is greater than a critical value, the force between the two beads is equal to zero.*

where a_{ij} represents the conservative soft potential parameter, r_{ij}=$|\mathbf{r_i} - \mathbf{r_j}|$ is the relative distance between two beads i and j, $\hat{\mathbf{r}}_{ij}$=$\frac{r_{ij}}{|r_{ij}|}$, and r_c is the cut-off radius, a characteristic length of the model. These repulsive potentials are much softer than the normal Lennard-Jones potentials, meaning that the overlap between two different beads during a single timestep is allowed. The choice of the a_{ij} parameter is not straightforward, and actually different values of this parameter, can be used to reproduce complex fluids. Its tuning becomes crucial to obtain the correct thermodynamic state. The first attempt to obtain results that were related to real systems, was the use of the fluctuations in liquid phase, physically represented by the compressibility of the system(Groot and Warren, 1997), κ:

$$\kappa^{-1} = \frac{1}{nk_BT\kappa_T} = \frac{1}{k_BT}\left(\frac{\partial P}{\partial n}\right)_T \tag{3.39}$$

where n is the molecular density, k_B is the Boltzmann constant, T is the temperature, P is the pressure and κ_T is the isothermal compressibility of the fluid. Water, at room temperature, was used as a reference and the non-dimensional value of κ^{-1} was found to be equal to 15.9835. When the density of the DPD system is higher than two particles per unit volume, a good approximation for the evaluation of the pressure can be given

by:

$$P_{DPD} = \rho_{DPD} k_B T + \alpha_{DPD} a_{ii} \rho_{DPD}^2 \tag{3.40}$$

where, ρ is the number of beads per unit volume, α is a tuning parameter equals to 0.101, and a_{ii} is the repulsive coefficient. Following from the equation of κ^{-1}, the compressibility of a fluid can be directly related to the repulsive coefficient:

$$\kappa^{-1} = 1 + 2\frac{\alpha_{DPD} a_{ii} \rho_{DPD}}{k_B T}, \tag{3.41}$$

resulting for water in:

$$a_{ii} = \frac{75 k_B T}{\rho_{DPD}}. \tag{3.42}$$

Simulations of different species, for example liquid-liquid systems, are extremely important in the chemical industry and in order to describe different chemical components, a correction to the a_{ij} parameter is needed. Polymers can be also described with a similar approach, where each bead has a specific value of the repulsive coefficient, based on the addition of an extra repulsion contribution, Δa. These extra contributions are directly derived from the Flory-Huggins χ-parameters, and they are physically representative of the solubility of one component into another. When the density of a DPD system is set equal to 3 (meaning that there are three beads for unit volume), the repulsive term between two species i and j is:

$$a_{ij} \approx a_{ii} + 3.27\chi_{ij} \tag{3.43}$$

In fact, the variation of the repulsive coefficient, according to the solubility of one component into another, allows the simulation of many species that reproduce the behaviour of chemical compounds.

The dissipative force is described as follows:

$$\mathbf{F_{ij}}^D = -\gamma_{ij} w^D(r_{ij})(\hat{\mathbf{r}}_{\mathbf{ij}} \cdot \mathbf{v_{ij}})\hat{\mathbf{r}}_{\mathbf{ij}} \tag{3.44}$$

where γ represents the dissipative coefficient acting as an artificial drag on the beads, w^D is a weight function that defines the maximum range of application of the force and $\mathbf{v_{ij}} = \mathbf{v_i} - \mathbf{v_j}$ is the relative velocity between two beads i and j. Its dependence on the velocity of the beads allows DPD to act as a thermostat in regulating the temperature of the system. The physical interpretation of the product between the velocity and the distance between two beads is the following: if this value is positive, a viscous force is exerted against particle i, and because of this, i moves apart from j.

If only dissipative and conservative forces were present, the system would simply freeze in its state. To solve this problem, continuous kicks to keep the particles in thermal motion are provided by a stochastic force, that can be described as follows:

$$\mathbf{F_{ij}}^R = \sigma_{ij} w^R(r_{ij}) \zeta_{ij} \Delta t^{-1/2} \hat{\mathbf{r}}_{\mathbf{ij}}, \tag{3.45}$$

where σ is the stochastic coefficient, w^R is again a weight function, ζ_{ij} is a random fluctuating variable with zero mean and unitary variance and Δt is the simulation timestep. Also, $\zeta_{ij} = \zeta_{ji}$ ensures that the total momentum is conserved, that is a fundamental aspect of this technique (Español and Warren, 2017). The weight functions and the stochastic and dissipative coefficients are linked through the fluctuation-dissipation theorem (Kubo, 1957) as follows:

$$w^D(\mathbf{r_{ij}}) = [w^R(r_{ij})^2] = \begin{cases} (1 - \frac{r_{ij}}{r_c})^2 & r_{ij} < r_c \\ 0 & r_{ij} > r_c \end{cases} \tag{3.46}$$

$$\sigma^2 = 2\gamma_{ij} k_B T, \tag{3.47}$$

where k_B is the Boltzmann constant and T is the temperature of the system. This last equation clarifies that the choice of one of the interaction parameter implies that the other is already defined.

Starting from the Newton's equations of motion, we now obtained a set of Langevin equations:

$$\mathrm{d}\mathbf{v_{ij}} = \left[\sum_{i \neq j} F_{ij}^C(r_{ij}) + \sum_{i \neq j} -\gamma_{ij} w^D(r_{ij}) (\hat{\mathbf{r}}_{\mathbf{ij}} \cdot \mathbf{v_{ij}}) \hat{\mathbf{r}}_{\mathbf{ij}} \right] \mathrm{d}t + \sum_{i \neq j} \sigma_{ij} w^R(r_{ij}) \hat{\mathbf{r}}_{\mathbf{ij}} \mathrm{d}W_{ij} \tag{3.48}$$

where $dW_{ij} = dW_{ji}$ are independent increments of the Wiener process.

DPD Polymeric Chains

Chains of beads used to represent for example polymers can be simulated by using DPD by adding an extra set of interactions to non-bonded ones. These bonded interactions can be described using different models. Bonded interactions are needed to maintain the topology of the polymer chains, meaning that a force is acting between two connected beads, preventing them to move apart from each other. In this work, two types of bonded potentials have been investigated: harmonic and finite-extensible nonlinear-elastic (FENE) potentials (Kremer and Grest, 1990). The harmonic potential is described as follows:

$$E_{harm} = \kappa_{ijharm} (r_{ij} - r_e)^2, \tag{3.49}$$

where κ_{ijharm} is the harmonic constant and r_e is the equilibrium distance between two connected beads, while the FENE potential is expressed by:

$$E_{FENE} = -\kappa_{ijFENE} r_e^2 \ln \left[1 - \left(\frac{r_{ij}}{r_e} \right)^2 \right], \tag{3.50}$$

where κ_{ijFENE} is the FENE bond constant, r_e is the equilibrium distance and r_{ij} is the distance between two beads. FENE potential is commonly used to prevent particles over-elongation of polymeric chains, when strong shear rates are applied on the systems. The value of the energy grows to infinite, while using harmonic potential, it could be possible that two beads belonging to the same chain, move further than the equilibrium distance.

DPD Scaling Factors

In order to describe a system, composed by identical particles, a coarse-graining level must be introduced (N_g). This number specifies how many real, physical particles or atoms or molecules are clustered into one single bead. According to this number, a set of units, reproducing equilibrium properties, can be retrieved and DPD quantities scaled to physical values. It is important to highlight that fundamental quantities are reported in DPD units and set equal to one. For example, a mass, m, a length, r_c, and an energy, k_BT, can be used as a conversion set to retrieve equilibrium properties of a system. If these three values are used, the characteristic DPD time, τ, becomes equal to (Groot and Madden, 1998):

$$\tau = r_c\sqrt{\frac{m}{k_BT}} \tag{3.51}$$

In the original DPD formulation, beads of different species are similar in shape (spherical) and volume. Usually, a certain number of water molecules (from three to six) is selected as benchmark. Other components then are clustered keeping in mind that each bead must contain similar volume occupied by water beads. Also, energy is not conserved, meaning that energy transport cannot be simulated if a corrected version of the DPD, Energy DPD is used instead. Marsh et al., 1997 proved that the dissipative forces acting on DPD beads, which depend on the thermal velocity, act as a thermostat on the system, meaning that temperature gradients in the system are not allowed. Also, the energy dissipated by those forces is invested into increasing the internal energy of the clusters (Español and Revenga, 2003). The calculation of the transport coefficient of a DPD system is non-trivial. As a practical example, one way to obtain the viscosity can then be by using the following equation:

$$\mu_{DPD} = -\frac{P_{xy}}{\dot{\gamma}}, \tag{3.52}$$

where $\dot{\gamma}$ is the imposed shear rate, while P_{xy} is the non-zero, xy non-diagonal component of the stress tensor. This technique can be used on a system where Lees-Edward boundary conditions (LEBC) are already implemented (see section 5.2.3 for LEBC description). In such a way, it is possible to reproduce and understand the behaviour of a bulk portion of the system where continuous stress is applied on a single element of fluid. But other techniques have been provided by Kubo, 1957

to obtain not only the viscosity but other transport coefficients (i.e. Green-Kubo formulation). However, the calculation of transport coefficients for DPD fluids may lead erroneous observations. One of the problems, present in the standard version of the DPD algorithm, is linked to the relation between the hydrodynamic interactions and the timescale for diffusion, namely the Schmidt number (Sc):

$$\mathrm{Sc} = \frac{\nu}{D} \tag{3.53}$$

The value obtained for a fluid representing water, using DPD, was in the order of magnitude of the unity, meaning that the mass is diffusing as fast as momentum, which is in constrast with experimental observations. This result is physically wrong, and the correct value should be in the order of magnitude of 10^3. Since Sc number is proportional to γ^2, by varying this parameter, an higher Sc can be achieved. This means that the evaluation of the correct transport coefficients should pass through the tuning of the γ parameter together with a_{ij}.

A final remark regards the conversion between DPD and real units. In order to compare the results of DPD simulations with experiments, it is necessary to define a conversion benchmark, such as a set of values representative of physical quantities. Having in mind that DPD beads are indeed a group of different clusters of atoms/molecules with the same size and weight, three variables can be used to define one possible conversion set (i.e. a length, a mass and a kinetic energy). The length, defined as a cut-off radius, represents the maximum level of interaction between DPD beads, the mass represents the number of particles clustered into one bead and the kinetic energy is an indicator of the thermal velocity of the beads. DPD simulations are performed using these parameters normalized to unity. If this set is fixed, all the remaining parameters can be obtained by their combination. An example of this conversion was reported by Groot and Rabone, 2001, and when the N_c, coarse-graining number is set equal to three:

$$r_{cut} = (3\rho_{DPD} V_{molecule})^{1/3} = 6.463\text{Å}, \tag{3.54}$$

where the cut-off length was obtained by assuming that each bead contains three molecules of water and the volume of one molecule of water is assumed to be:

$$V_{molecule} = 3 \times 10^{-29}\mathrm{m}^3, \tag{3.55}$$

$$m_{bead} = 3 \cdot m_{molecule} = 8.967 \cdot 10^{-26}\mathrm{kg}, \tag{3.56}$$

hence the mass of a single bead can be obtained by defining the number of molecules clustered in it. Finally, the energy or the velocity of the beads in the system can be used to derive the time unit. If the thermal velocity if chosen, the following approach can be selected at equilibrium:

$$v_{bead} = \sqrt{\frac{3k_BT}{m}} \tag{3.57}$$

While if the temperature is chosen as a standard reference, the value of 298 K can be used to calculate the value of the DPD energy and it is independent from the coarse-graining number:

$$\epsilon_{DPD} = k_BT = 4.112 \cdot 10^{-21}\text{J}, \tag{3.58}$$

Although this conversion set of parameters is producing consistent results in evaluating equilibrium properties, the same does not apply in non-equilibrium conditions. The conversion of DPD values into real physical units, according to the equilibrium conversion set, could produce unrealistic values for non-equilibrium quantities, such as for example the conversion between real shear rate and the DPD one (similarly to what happens in non-equilibrium AAMD). It is also necessary to say that, given this set of parameters and types of interaction, chain-crossing is allowed. This could also bring to deviations from real, physical quantities. Despite all these limitations we think the present analysis is useful and can generate interesting results.

3.3 Computational Fluid Dynamics

Multiphase and mixing processes can be modelled at the industrial scale by using CFD (Aubin et al., 2004; Coroneo et al., 2011; Nere et al., 2003; Sahu et al., 1999; Sommerfeld and Decker, 2004). CFD can be used at industrial level for length and timescales that cannot be accessed by the models previously discussed. The capabilities of CFD scale with both the available computing power (tech limit) and the progresses made in the derivation of more and more accurate models, for example coming from MD, CG but also experiments (knowledge limit). Starting from single flows in common geometries and validations against empirical solutions, it also became possible to tackle problems related to multiphase and multifluid systems, by using this computational tool, even if strong approximations and closure models were needed.

When assessing multiphase systems, three main categories can be used to describe a system depending on the desired output and on the required level of detail that is requested. These are: Eulerian-Eulerian (E-E) , Eulerian-Lagrangian (E-L) and Direct Numerical Simulation (DNS). Moving from E-E to DNS it is possible to move down in the geometrical scale, but the computational time is increased.

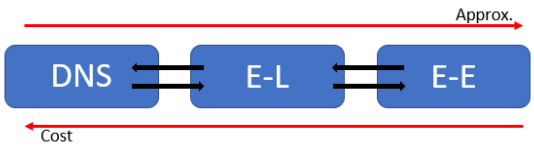


Figure 3.4. *Three scales are used in continuum modelling. The level of accuracy moves from DNS (most accurate) to E-E (least accurate) but the computational cost moves in the opposite direction.*

In the E-E model (or two- multi- fluid model), all the phases are treated as compenetrating media and solved as segregated fluid phases. For each phase, a continuity equation (i.e. mass balance) together with a momentum and energy balance are solved. In this model, it is important to define a control volume bigger than the molecular (or particle) scale but smaller with the equipment scale, in order to average the properties of the continua over a fixed volume. The main disadvantage of E-E models lies in the impossibility of tracking single particles, hence only average fields can be obtained. For such purpose strong approximations and models based on empirical evidences can be provided .

To overcome these problems, a new method was introduced: the Lagrangian-Eulerian formulation or Discrete Element Model (DEM). In 1979, Cundall and Strack, 1979 proposed this new methodology where one phase is treated as a continuum while the second one is still a disperse phase, but each particle or cloud or particles can be tracked. Transport equations for properties related to the continuum phase are solved in a segregate manner in respect to the disperse phase. On the other side,

trajectories and collisions are computed for each single discrete element. Different coupling techniques between the two phases have been proposed, depending on the interaction level between Eulerian and Lagrangian phases. Phases can be fully coupled, i.e. they both influence each other, partially coupled or fully decoupled. Many models have been introduced in order to describe the interactions between particles (hard spheres, soft spheres, etc.) which becomes the crucial part of E-L model. The main disadvantage in using L-E models is the computational time that is used to simulate the interaction between particles and the calculation of the trajectories of each single body. This means that systems containing a big number of bodies cannot be simulated using this approach. The last technique which was introduced in CFD that contains an even higher level of information is called Direct Numerical Simulation (Tryggvason et al., 2001). Using this model, it is possible to solve directly the interface of each particles and quantify the contribute of different forces acting on it. More than thirty cells must be used to describe each single particle, meaning that a short spatial resolution can be explored. Volume of fluid, level-set and front-tracking methods, represent some of the most important applications of such method. For all of them, just one set of equation is solved together with an extra function that tracks the evolution of the interface between the phases (Brennen, 2013; Osher and Fedkiw, 2001).

In this work, E-E model is used to model emulsions.

3.3.1 From the Boltzmann equation to the Navier-Stokes equation

In this section the governing equations are obtained starting from the Boltzmann equation. Some concepts must be re-introduced: the probability of finding a particle at time t, in the position $\boldsymbol{x}$ and with velocity $\boldsymbol{v}$ is described by the probability density function $f(t,\boldsymbol{x},\boldsymbol{v})$. If we consider the quantity $f(t,\boldsymbol{x},\boldsymbol{v})dxdv$, it represent the mass of fluid contained within a cube of dimension $dx = dxdydz$, with velocity $dv = dv_x dv_y dv_z$. Average quantities, such as density or average velocity, in time and space can be derived by the integration of the probability distribution function over the space of all the possible velocities:

$$\rho(t,\mathbf{r}) = \int\int\int_{-\infty}^{+\infty} f(t,\mathbf{x},\mathbf{v})dv_x dv_y dv_z, \tag{3.59}$$

$$\overline{v}_i(t,\mathbf{r}) = \frac{1}{\rho}\int\int\int_{-\infty}^{+\infty} v_i f(t,\mathbf{r},\mathbf{v})dv_x dv_y dv_z, \tag{3.60}$$

The Boltzmann equation can be used to describe the evolution of the system:

$$\frac{\partial f}{\partial t} + v_i\frac{\partial f}{\partial x_i} - g\frac{\partial f}{\partial v_3} = \mathscr{C}(f), \tag{3.61}$$

where summation is represented by repeated indices, g is gravitational force and v_3 is the axis of the force of gravity but in the opposite direction. This equations means

that the probability distribution function moves in space because of the velocity of the particles and this velocity is altered by the gravity and redistributed by the collisions. An assumption must be made at this point in order to recover the Navier-Stokes equation, that describes the transport of momentum within the continuum framework. The effect of the collision integral, $\mathscr{C}$ is to move f towards an equilibrium distribution. The equilibrium distribution can be written in the form of the Maxwell-Boltzmann distribution:

$$f^{eq}(t, \mathbf{x}, \mathbf{v}) = \frac{\rho(t, \mathbf{x})}{2(\pi)^{3/2} V^3(t, \mathbf{x})} exp\left(-\frac{|\mathbf{v} - \overline{\mathbf{v}}(t, \mathbf{x})|^2}{2V^2(t, \mathbf{x})}\right) \tag{3.62}$$

where V is the standard deviation, and the collision term can be approximated as:

$$\mathscr{C}(f) = \frac{1}{\tau}(f^{eq} - f). \tag{3.63}$$

such that the Eq. 3.65 becomes:

$$\frac{\partial f}{\partial t} + v_i \frac{\partial f}{\partial x_i} = \frac{1}{\tau}(f^{eq} - f) + \mathbf{g}\frac{\partial f}{\partial v_3}. \tag{3.64}$$

Mass can be obtained by integrating in the velocity space Eq. 3.64, or calculating the moment of order zero:

$$\int\int\int_{-\infty}^{+\infty} \left\{\frac{\partial f}{\partial t} + v_i \frac{\partial f}{\partial x_i}\right\} d\mathbf{v} = \int\int\int_{-\infty}^{+\infty} \left\{\frac{1}{\tau}(f^{eq} - f) + g\frac{\partial f}{\partial v_3}\right\} d\mathbf{v} \tag{3.65}$$

according to the definition of density derived in Eq. 3.59, assuming that temporal and spatial derivates can be taken outside the integrals because they do not depend on the velocity, and the collisional term vanishes because mass is conserved in each collision, the continuity equation can be obtained:

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}(\rho v_i) = 0 \tag{3.66}$$

Similarly, starting from the moment of order one of the Boltzmann equation, it is possible to derive the Navier-Stokes equation, that describes the transport of momentum:

$$\int\int\int_{-\infty}^{+\infty} \left\{u_j \frac{\partial f}{\partial t} + v_i \frac{\partial f}{\partial x_i}\right\} d\mathbf{v} = \int\int\int_{-\infty}^{+\infty} v_j \left\{\frac{1}{\tau}(f^{eq} - f) + g\frac{\partial f}{\partial v_3}\right\} d\mathbf{v} \tag{3.67}$$

by assuming that the collisions conserve momentum and the overline represents the average value, Eq 3.67 becomes:

$$\frac{\partial}{\partial t}(\rho\overline{v}_j) + \frac{\partial}{\partial x_i}(\rho\overline{v}_i\overline{v}_j) = \frac{\partial\sigma_{ij}}{\partial x_i} - \rho g\delta_{j3} \tag{3.68}$$

where

$$\sigma_{ij} = \int\int\int (v_i - \overline{v}_i)(v_j - \overline{v}_j) f\, d\mathbf{v}. \tag{3.69}$$

for a single phase Newtonian fluid, it is therefore possible to write the Navier-Stokes equation in its complete form:

$$\frac{\partial\rho v_i}{\partial t} + \left[\frac{\partial}{\partial x_i}(\rho v_i v_j)\right] = -\frac{\partial p_i}{\partial x_i} + \mu\frac{\partial^2}{\partial x_i^2}v_i + \rho\mathbf{g} \tag{3.70}$$

3.3.2 Two-Fluid Model

In the E-E formalism the most common model is the Two-Fluid model (TFM) introduced by Ishii and Mishima (Ishii et al., 1982), in which volume fraction and average velocity fields are used to describe the evolution of the system. This model can be used to describe complex multiphase systems (i.e. a continuum and a disperse phase) where properties are defined in terms of average quantities. Single identities, as per the disperse phase, are merged and described as fields. Thanks to TFM, the evolution of these fields is tracked and average quantities are retrieved. Two sets of governing equations are written for the disperse and continuum phase and since these two can interact, extra terms are introduced according to the level of interaction. A continuity equation for the disperse phase, reads as follows:

$$\frac{\partial(\alpha_i^d\rho_i^d)}{\partial t} + \frac{\partial}{\partial x_i}(\alpha_i^d\rho_i^d v_i^d) = 0, \tag{3.71}$$

where α_i^d, ρ_i^d , and v_i^d are respectively the volume fraction, the density and the velocity of the disperse phase, while t is the time. Right-hand side is null because mass is conserved and no source terms are present. A momentum balance equation is solved for the continuous phase:

$$\frac{\partial\alpha_i^c\rho_i^c v_i^c}{\partial t} + \frac{\partial}{\partial x_i}\alpha_i^c\rho_i^c(v_i^c v_i^c) + \frac{\partial}{\partial x_i}(\alpha_c\tau^c) + \frac{\partial}{\partial x_i}(\alpha_c R_i^c) = -\alpha_i^c\frac{\partial p}{\partial x_i} + \alpha_i^c\rho_i^c g_i - \boldsymbol{M}_i^c \tag{3.72}$$

and another one is solved for the disperse phase:

$$\frac{\partial\alpha_i^d\rho_i^d v_i^d}{\partial t} + \frac{\partial}{\partial x_i}\alpha_i^d\rho_i^d(v_i^d v_i^d) + \frac{\partial}{\partial x_i}(\alpha_i^d\tau^d) + \frac{\partial}{\partial x_i}(\alpha_i^d\mathbf{R}_\mathbf{i}^\mathbf{d}) = -\alpha_i^d\frac{\partial p}{\partial x_i} + \alpha_i^d\rho_i^d g_i + \boldsymbol{M}_i^d \tag{3.73}$$

Where v_i^c and v_i^d are the velocity of the continuous and disperse phase, τ^c and τ^d are the viscous stress tensors, and $\mathbf{R^c}$ and $\mathbf{R^d}$,the Reynolds stress tensors, p is the pressure, g is the gravity, and $\mathbf{M^c}$ and $\mathbf{M^d}$ describe the interfacial forces. In particular, $\mathbf{M^d}$ can be expressed by:

$$\mathbf{M_d} = \alpha_d \alpha_c \left(\frac{3}{4} C_D \frac{\rho_d}{d_{32}} |\mathbf{v_r}| \right) \mathbf{v_r}, \tag{3.74}$$

where $\mathbf{v_r} = \mathbf{v_c} - \mathbf{v_d}$, d_{32} is the Sauter diameter, while the drag force coefficient C_D can be described according to Schiller and Naumann correlation:

$$C_D = \left\{ \begin{array}{cc} \frac{24}{\mathrm{Re}}(1 - 0.15\mathrm{Re}^{0.687}) & \mathrm{Re} \leq 1000 \\ 0.44 & \mathrm{Re} > 1000 \end{array} \right\}. \tag{3.75}$$

where Re is the local Reynolds number ($\mathrm{Re} = \frac{\rho_c |\mathbf{v_r}| d_{32}}{\mu_c}$). Different models for the interfacial stresses were tested and results reported in the Results session. The turbulence plays an important role in mixing problems. In particular, for this model, we used Reynolds-averaged Navier-Stokes equation (RANS) and tested different turbulence models.

3.3.3 Turbulence models

Turbulence models must be included in the correct formulation of flow fields, when the Reynolds number is sufficiently high. In particular, in the part related to the mixing of this work, different turbulence models have been tested, since high velocities are obtained in some regions of the systems. Reynolds-Averaged Navier-Stokes (RANS) models have been used and compared against each other. These models describe from two to seven different sets of transport equations for the turbulent variables. In particular, $\kappa - \epsilon$, $\kappa - \omega$ and Reynolds Stress models were used. Only the governing equations that are solved for each model are reported.

Standard $\kappa - \epsilon$ Model

In $\kappa - \epsilon$ model, two transport equations are solved for the turbulent kinetic energy and the turbulent dissipation rate:

$$\frac{\partial}{\partial t}(\rho\kappa) + \frac{\partial}{\partial x_i}(\rho\kappa v_i) = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_\kappa} \right) \frac{\partial \kappa}{\partial x_j} \right] + G_k + G_b - \rho\epsilon_{turb} - Y_M + S_\kappa \tag{3.76}$$

and

$$\frac{\partial}{\partial t}(\rho\epsilon_{turb})+\frac{\partial}{\partial x_i}(\rho\epsilon_{turb}v_i) = \frac{\partial}{\partial x_j}\left[\left(\mu+\frac{\mu_t}{\sigma_{\epsilon_{turb}}}\right)\frac{\partial\epsilon_{turb}}{\partial x_j}\right]+C_{1\epsilon_{turb}}\frac{\epsilon}{\kappa}(G_\kappa+C_3G_b)-C_{2\epsilon_{turb}}\rho\frac{\epsilon_{turb}^2}{\kappa}+S_\epsilon. \tag{3.77}$$

where G_k and G_b represent the generation of turbulent kinetic energy because of the mean velocity gradients and buoyancy, Y_M is given by $2\rho\epsilon_{turb}\frac{\kappa}{\gamma RT}$, being $\sqrt{\gamma RT}$ the speed of sound, σ_k and $\sigma_{\epsilon_{turb}}$ are the turbulent Prandtl numbers and S are external sources. The turbulent viscosity can be obtained with the following equation:

$$\mu_t = \rho C_\mu\frac{\kappa^2}{\epsilon_{turb}} \tag{3.78}$$

The constants have values $C_{1\epsilon_{turb}} = 1.44, C_{2\epsilon_{turb}} = 1.92, C_\mu = 0.09, \sigma_\kappa = 1.0$ and $\sigma_{\epsilon_{turb}} = 1.3$

$\kappa-\omega$ Shear Stress Transport Model

In $\kappa-\omega$ - SST model the following equations are solved:

$$\frac{\partial}{\partial t}(\rho\kappa)+\frac{\partial}{\partial x_i}(\rho\kappa v_i) = \frac{\partial}{\partial x_j}\left(\Gamma_\kappa\frac{\partial\kappa}{\partial x_j}\right)+G_k-Y_k+S_k, \tag{3.79}$$

$$\frac{\partial}{\partial t}(\rho\omega)+\frac{\partial}{\partial x_i}(\rho\omega\mathbf{v_i}) = \frac{\partial}{\partial x_j}\left(\Gamma_\omega\frac{\partial\omega}{\partial x_j}\right)+G_\omega-Y_\omega+D_\omega+S_\omega, \tag{3.80}$$

where the single terms represent:

$$\Gamma_i = \mu+\frac{\mu_t}{\sigma_i}, \tag{3.81}$$

$$\mu_t = \frac{\rho\kappa}{\omega}\frac{1}{max\left[\frac{1}{\alpha*},\frac{SF_2}{\alpha_1\omega}\right]}, \tag{3.82}$$

$$\sigma_i = \frac{1}{\frac{F_1}{\sigma i,1}+\frac{(1-F_1)}{\sigma_{i,2}}}, \tag{3.83}$$

$$G_\kappa = min(G_\kappa, 10\rho\beta^*\kappa\omega), \tag{3.84}$$

$$G_\omega = \frac{\alpha}{\nu_t}G_\kappa, \tag{3.85}$$

$$Y_\kappa = \rho\beta^*\kappa\omega, \tag{3.86}$$

$$Y_\omega = \rho\beta\omega^2, \tag{3.87}$$

$$D_\omega = 2(1-F_1)\rho\omega_{\omega,2}\frac{1}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial\omega}{\partial x_j}, \tag{3.88}$$

where S is the strain rate, F_i are blending functions and all the remaining terms are constants of the model.

Reynolds Stress model

This model is based on the idea of solving additional equations for the components of the Reynolds stress tensor, the transport equation for the Reynolds stresses read as follows (Andersson et al., 2011)

$$\underbrace{\frac{\partial}{\partial t}(\overline{\rho v_i' v_j'})}_{\text{Time Derivative}} + \underbrace{\frac{\partial}{\partial x_k}(\rho v_k \overline{v_i' v_j'})}_{\text{Convective Term}} = \underbrace{-\frac{\partial}{\partial x_k}\left[\overline{\rho v_i' v_j' v_k'} + \overline{p(\delta_{kj} u_i' + \delta_{ik} u_j')}\right]}_{\text{Turb. Diffusion}}$$
$$+ \underbrace{\frac{\partial}{\partial x_k}\left[\mu \frac{\partial}{x_k}(\overline{(v_i' v_j')}\right]}_{\text{Molecular Diffusion}} - \underbrace{\rho\left(\overline{v_i' v_k'}\frac{\partial v_j}{\partial x_k} + \overline{v_j' v_k'}\frac{\partial v_i}{x_k}\right)}_{\text{Stress Production}} - \underbrace{\rho\beta(g_i\overline{v_j'\theta} + g_j\overline{v_i'\theta})}_{\text{Buoyancy Production}}$$
$$+ \underbrace{\overline{p\left(\frac{\partial v_i'}{\partial x_j} + \frac{\partial u_j'}{\partial x_i}\right)}}_{\text{Pressure Strain}} - \underbrace{2\mu\overline{\frac{\partial v_i'}{\partial x_k}\frac{\partial v_j'}{\partial x_k}}}_{\text{Dissipation}} - \underbrace{2\rho\omega_k(\overline{v_j' v_m'}\epsilon_{ikm} + \overline{v_i' v_m'}\epsilon_{jkm})}_{\text{Production by System Rotation}} + \underbrace{S_{ext}}_{\text{Extra Sources}} \tag{3.89}$$

being the overline the Reynolds average.

3.3.4 Population Balance Equation

In TFM, interaction terms need the definition of the Sauter diameter (or d_{32} which is the equivalent diameter of a sphere with the ratio between volume and surface equals to the droplets'one). The population balance model (PBM) can be used to describe the evolution of a population of elements of disperse phase (e.g. droplets) present within one system (Marchisio and Fox, 2013), and in particular to track the evolution of d_{32}. It was originally formulated by Smoluchowski in 1916 (Smoluchowski, 1916), then improved between 1985 and 2000 by Ramakrishna and Mahoney (Ramakrishna, 2000). The simplest formulation of the model follows:

$$\frac{\partial n(\xi;\mathbf{r},t)}{\partial t} + \mathbf{v_i^d}\frac{\partial n(\xi;\mathbf{x},t)}{\partial r_i} - \frac{\partial}{\partial x_i}\left[(\Gamma+\Gamma_t)\frac{\partial n(\xi,\mathbf{x},t)}{\partial x_i}\right] = -\frac{\partial}{\partial \xi_j}\left[n(\xi;t)\zeta_j\right] + h(\xi;t), \tag{3.90}$$

Where ξ is a internal coordinate, $\mathbf{r}$ is an external coordinate of the system, $\mathbf{v_d}$ is the Reynolds-average velocity, n is the number of discrete entities in a unitary volume, Γ, Γ_t are the diffusion and the turbulent diffusion coefficients, and h describes the introduction of new entities into the system. If we assume that only one coordinate is tracked, for example an internal one-dimensional coordinate, the previous equation may be simplified as follows:

$$\frac{\partial n(\xi)}{\partial t} + \nabla\cdot(\mathbf{v_d} n(\xi)) = \mathcal{S}(\xi), \tag{3.91}$$

where $n(\xi)$ describes the droplet size distribution and ξ is the droplet diameter. The source terms contain all the information related to the phenomena that involve the evolution of the droplets in time, such as coalescence, breakage, and nucleation. Depending on the system of interest, these phenomena have to be included or removed from the model. Droplets can coalesce and break up during the mixing because of different aspects. In order to take into account all possible phenomena involved, the source term present in the Equation 3.91, can be re-written as follows:

$$\begin{aligned} \mathcal{S}(\xi) = \frac{\xi^2}{2}\int_0^{\xi} \frac{a(\xi^3-\xi'^3,\xi')}{(\xi^3-\xi'^3)^{2/3}} n((\xi^3-\xi'^3)^{1/3}) n(\xi')\mathrm{d}\xi' \\ -n(\xi)\int_0^{\infty} a(\xi,\xi')n(\xi')\mathrm{d}\xi' + \int_{\xi}^{\infty} g(\xi')\beta(\xi|\xi')n(\xi')\mathrm{d}\xi' - g(\xi)n(\xi), \end{aligned} \tag{3.92}$$

where ξ and ξ' and are two sizes (initial and final) of the droplet diameter, $a(\xi,\xi')$ is coalescence kernel, $g(\xi')$ is the breakage kernel and $\beta(\xi|\xi')$ is the daughter distribution function, that describes how many droplets form after every break up event. If we can assume that in cases where the disperse phase is low but also surfactant is present, we can ignore the coalescence term, and the Eq. 3.91, can be re-written as follows:

$$\frac{\partial n(d)}{\partial t} + \nabla\cdot(n(d)\mathbf{v_d}) = \int_a^{\infty} \beta(d,d')g(d')dd' - g(d)n(d), \tag{3.93}$$

The ways in which PBE can be solved are many and one example is provided by the Quadrature Method of Moments (QMOM).

3.3.5 Breakage Models

Breakage phenomenon has been described by many authors and it consists in the rupture of a drop, suspended in a continuous phase, into smaller droplets, because of instantaneous stresses overcoming the stabilizing effects of interfacial tension and drop viscosity (Gao et al., 2016; D. Li et al., 2017). In mixing processes, where the final size of the droplets generated in the mixture is important for stabilizing effects but also for the prediction of the viscosity, it becomes fundamental to understand the link between the turbulence induced by the impellers, locally surrounded by high shear regions, and the frequency of breakage. Even if other relevant phenomena (such as coalescence) should be taken into account when describing the evolution of the DSD, in cases of low volume fraction and presence of surfactant, they can be neglected. The first studies regarding the breakage phenomenon were started by Kolmogorov and Hinze in 1949 – 1955. In the first models, where there were no fluctuations of the turbulent dissipation rate, an expression to obtain the maximum stable drop that can be obtained by turbulent fluctuations was obtained:

$$d^0_{max} = \sigma_t^{0.6} C_x < \epsilon_{turb} >^{-0.4} \rho_c^{-0.6} \tag{3.94}$$

Where the relevant parameters that affect the breakage phenomenon started to appear, such as the interfacial tension, the turbulent dissipation rate and the density of the continuous phase. This was the initial step through the development of new and more sophisticated models, such the one proposed by Coulaloglou and Tavlarides (CT) (Coulaloglou and Tavlarides, 1977), Alopaeus and Laakkonen(Laakkonen et al., 2007), and Baldyga and Podgorska(Baldyga and Podgórska, 1998). Many models have been proposed during the years but most of them lie on empirical observations. Models for liquid-liquid, gas-liquid, solid-liquid systems depend on different parameters or are finely tuned by playing with the values of the constants. A short description of the models used in this work, related to liquid-liquid systems, is now reported.

Coulaloglou and Tavlarides Kernel

In CT kernel, the breakage is driven by the transmission of the turbulent kinetic energy, coming from the eddies of the continuous phase, to the surface of the droplet. When the transmitted energy is greater than the surface energy of the droplet, it breaks up into daughter droplets. Two main assumptions must be satisfied in order to have good predictions with CT kernel, the hypothesis of local isotropy and the diameter must belong to the inertial sub-range.

$$g_{CT}(d) = C_1 \frac{\epsilon_{turb}^{1/3}}{d^{2/3}} \exp\left(-C2 \frac{\sigma_t}{\rho_c \epsilon_{turb}^{2/3} d^{5/3}}\right), \tag{3.95}$$

The terms C_1 and C_2 are experimental constants obtained by fitting, ϵ_{turb} is the turbulent kinetic energy, σ_t is the interfacial tensions between continuous and disperse phase, d is the droplet diameter, ρ is the density of the continuous phase. The values of the constant C_1 and C_2 are derived from experiments and for this specific case they were set equal to 0.00481 and 0.08.

Alopaeus and Laakkonen (LA) Kernel

LA kernel was derived from Narsimhan kernel that was introduced for liquid-liquid dispersion. The original model proposed by Narshiman:

$$g_{NA}(d) = C_3 erfc\left(3.5 \left(\frac{d_p}{d_p*}\right)^{-5/6}\right), \tag{3.96}$$

It was corrected by Alopaeus first, by adding the dependence on the turbulence dissipation rate to the eddy collision frequency, and then together with Laakkonen, to include the effect of different viscosities for the disperse and continuous phases. Indeed, the LA kernel includes stabilizing effects that are intrinsically produced by the

viscosity of the disperse phase. In cases where the viscosities of the mixed components are different, it becomes important to introduce this extra stabilizing contribute in the calculation of the breakage frequency, that acts as an extra force that opposes to the breakage. The Alopaeus-Laakkonen kernel still contains empirical constant that were tuned on many different silicone-oil water emulsions. However, the value of these constants was obtained when the viscosity of the disperse phase was not too high. The kernel contains an error function complementary, that goes to zero, as the two groups enclosed by the function become bigger (i.e. droplet size reduces with time). It depends on the turbulent dissipation energy of the system, mostly generated in the area close to the impeller, and on the equilibrium between interfacial force and viscous stress generated inside the droplet.

$$g_{LA}(d) = C_5 \epsilon_{turb}^{1/3} erfc\left(\sqrt{C_2 \frac{\sigma_t}{\rho_c \epsilon_{turb}^{2/3} d^{5/3}} + C_3 \frac{\mu_d}{\sqrt{\rho_c \rho_d} \epsilon_{turb}^{1/3} d^{4/3}}}\right), \tag{3.97}$$

where C_5, C_2, C_3 are empirical constants, ϵ_{turb} is the turbulence dissipation energy, σ_t is the interfacial tension, μ_d is the viscosity of the disperse phase, ρ_c, ρ_d are the densities of the continuous and the disperse phase, d is the diameter of the droplet, $erfc$ is the complementary error function ($erfc = 1 - erf$) that, as previously mentioned, goes to zero when droplets become too small (dimensions drop into the viscous subrange). One problem of the LA kernel is that the constant C_5 is not dimensionless, but it has the dimension of length to the minus-two-third, and a scaling value could be necessary for describing different cases. Also, C_3 and C_2, in some specific cases may need fine tuning, based on the simulated system.

Baldyga and Podgorska (BP) Kernel

The BP kernel is also known as multifractal kernel. It includes the intermittent nature of the turbulence in the description of the breakage phenomenon, namely short-lived velocity gradient are responsible for intermittent time evolution of turbulent properties, such as turbulent dissipation energy. The large fluctuations on the turbulent kinetic energy are considered as important phenomena especially in scaling up and down the processes, meaning that they must be carefully taken into account especially in problems related to the scaling from lab to industrial equipment when describing liquid-liquid dispersion. The BP kernel is able to capture the fine scale structure of turbulence, composed by areas where the velocity and the local shear is high, surrounded by nearly irrotational fluid. The breakage frequency can be obtained by:

$$g_{BP}(d) = C_g \sqrt{\ln\left(\frac{L}{d}\right)} \frac{\epsilon_{turb}^{1/3}}{d^{2/3}} \int_{\alpha_{min}}^{\alpha_x} \left(\frac{d}{L}\right)^{\frac{\alpha_{BP}+2-3f(\alpha_{BP})}{3}} \mathrm{d}\alpha_{BP}, \tag{3.98}$$

Where C_g is an empirical constant set equal to 0.0035 calculated by Bałdyga and Podgórska by fitting drop size distributions predicted using multifractal breakage model to the experimental results proposed by Konno et al., 1983 and it represents the rate of breakage in agitated tank with specific setups (dimensions, diameters...), α_{BP} is the multifractal exponent, d is the diameter of the droplet, $f(\alpha_{BP})$ describes the multi-fractal spectrum with a universal form, which can be described by the following polynomial function:

$$f(\alpha_{BP}) = a + b\alpha_{BP} + c\alpha_{BP}^2 + d\alpha_{BP}^3 + e\alpha_{BP}^4 + f\alpha_{BP}^5 + g\alpha_{BP}^6 + h\alpha_{BP}^7 + i\alpha_{BP}^8 \quad (3.99)$$

With a = -3.51, b = 18.721, c = -55.918, d = 120.9, e=-162.54, f=131.51, g = -62.572, h = 16.1, i = -1.7264; L is the integral turbulent length scale and can be calculated obtained by:

$$L = \frac{\frac{2}{3}k_{turb}^{\frac{3}{2}}}{\epsilon_{turb}} \quad (3.100)$$

Where k_{turb} is the turbulent kinetic energy. The introduction of L links the description of the breakage event to the scale of the system and in general, the kernel predicts slow breakage for very small droplets. It is important to underline that the BP kernel was developed for the inertial subrange, meaning that the droplet size must be smaller than the macro-scale (L has typically values in the order of magnitude as the impeller dimension) but greater than the Kolmogorov scale ($\eta_i = \frac{\nu^{3/4}}{\epsilon_{turb}^{1/4}}$). Below the Kolmogorov scale, these kernels predict zero breakage velocity (that is predicted when maximum stable diameter is reached) because the mechanisms in the viscous scale are different from the ones described by these kernels. Baldyga, Bourne, et al., 1995 noticed that the break-up in inertial subrange is a short duration process, smaller than the duration of a turbulent event. However, Baldyga and Podgorska also derived two kernels for droplets smaller than the Kolmogorov scale.

Multifractal exponent α_{BP}, represents the strength of the eddies. Eddies transfer energy to the droplet and their activity can be labelled by different values of α_{BP}. α_{min} the lower bound equals to 0.12 proposed by Meneveau and Sreenivasan, 1991, represents the spectrum of vigorous eddies that are able of breaking a droplet. while α_{min} and α_x, the upper bound represents the most vigorous eddies that can cause breakage and when a value of α_x approaches the value of α_{min}, the integral goes to zero, meaning that the emulsion is stable. When α_{BP} is smaller than α_{min} , as it is in the bulk, no breakage occurs. Values of α_x smaller than 1, represents violent but less frequent bursts events, meaning that quasi stable DSD are obtained after long agitation time. In the description of the kernel, different mechanisms that act against the breakage are also taken into account. In particular, high viscosity of the disperse adds extra stabilizing stress to the droplet, namely internal flow inside the droplet cause viscous stresses

that prevent deformation. This means that in computing the equilibrium between the different phenomena acting on the surface of the droplet, pressure fluctuations, internal viscous stress and interfacial stress must be considered:

$$\alpha_x = 3 \cdot \frac{\ln\left\{2\left[\frac{\beta_\mu C_x^{5/3}\mu_d}{\rho_c<\epsilon_{turb}>^{1/3}L^{1/3}d} + \sqrt{(\frac{\beta_\mu C_x^{5/3}\mu_d}{\rho_c<\epsilon_{turb}>^{1/3}L^{1/3}d})^2 + \frac{4C_x^{5/3}\sigma}{\rho_c<\epsilon_{turb}>^{2/3}L^{2/3}d}}\right]^{-1}\right\}}{\ln(\frac{L}{d})} \tag{3.101}$$

Where C_x describes the maximum quasi stable droplet and its value, obtained by Lagisetty et al., 1986, is equal to 0.23, μ_d is the viscosity of the disperse phase, and β_μ is calculated by:

$$\beta_\mu = \frac{\ln\left(\frac{x_\beta}{d}\right)_b}{\beta^* C_P C_x^{5/3}}, \tag{3.102}$$

Where the ratio $\left(\frac{x_\beta}{d}\right)_b$ is equal to 2, C_p = 1.4, and β^* is equal to 3. This is the complete form of the multifractal exponent that is able to handle also stabilizing effect due to high viscosity of the dispersed phase.

Chapter 4

Solution Algorithms

4.1 Introduction

In this Chapter, the solution algorithms for the models described in Chapter 3 are reported. In particular the first part is dedicated to algorithms for solving DPD and the second part is mostly focused on CFD models and the quadrature method of moments (QMOM) to solve the population balance equation.

4.2 Dissipative Particle Dynamics

In this part, the integration procedures used for solving DPD governing equations are presented. Particular focus is given to the Euler, the Leap-Frog and the Verlet Algorithm (Allen and Tildesley, 2017).

4.2.1 Euler Algorithm

The Euler integration scheme can be used to integrate the governing equations of DPD as follows:

$$\mathbf{x_i}(t + \Delta t) = \mathbf{x_i}(t) + \mathbf{v_i}\Delta t, \tag{4.1}$$

$$\mathbf{v_i}(t + \Delta t) = \mathbf{v_i}(t) + \mathbf{a_i}\Delta t, \tag{4.2}$$

$$\mathbf{F_i}(t + \Delta t) = \mathbf{F_i}(x(t + \Delta t), \mathbf{v}(t + \Delta t)), \tag{4.3}$$

where positions and velocities belonging to the particle i, are updated starting from their previous value in time. This algorithm is not commonly used because it does not result in reversible trajectories.

4.2.2 Leap Frog Algorithm

This integration method is also quite used in integrating MD and CG simulations. It can be expressed according to its 'kick-drift-kick' version:

$$\mathbf{v_{i+1/2}} = \mathbf{v_i} + \mathbf{a_i}\frac{\Delta t}{2}, \tag{4.4}$$

$$\mathbf{x_{i+1}} = \mathbf{x_i} + \mathbf{v_{i+1/2}}\Delta t, \tag{4.5}$$

$$\mathbf{v_{i+i}} = \mathbf{v_{i+1/2}} + \mathbf{a_{i+1}}\frac{\Delta t}{2} \tag{4.6}$$

Advantages of leap from algorithm are the time reversibility and the capability of conserving energy of dynamical systems.

4.2.3 Verlet Algorithm

Beads move according to a modified version of the Verlet algorithm, in which an extra stochastic term is introduced. The DPD version of the Verlet algorithm improves the capability of choosing bigger timesteps and obtaining better results compared to Euler algorithms. The modified Verlet algorithm reads as follows:

$$\mathbf{x_i}(t + \Delta t) = \mathbf{x_i}(\mathbf{t}) + \Delta t \mathbf{v_i}(t) + \frac{1}{2}(\Delta t)^2 \mathbf{a_i}(t) \tag{4.7}$$

$$\mathbf{v_i^*}(t + \Delta t) = \mathbf{v_i}(t) + \lambda_{Verlet}\Delta t \frac{\mathbf{F_i}(t)}{m}, \tag{4.8}$$

$$\mathbf{F_i}(t + \Delta t) = \mathbf{F_i}(\mathbf{x}(t + \Delta t), \mathbf{v_i^*}(t + \Delta t), \tag{4.9}$$

$$\mathbf{v_i}(t + \Delta t) = \mathbf{v_i}(t) + \frac{1}{2}\Delta t(\mathbf{F_i}(t) + \mathbf{F_i}(t + \Delta t)), \tag{4.10}$$

where $\mathbf{v_i^*}$ is a first approximation value of the velocity, that is used to calculate the forces, which also contain the velocity term acting between different beads. The extra coefficient λ takes into account stochastic effects and random fluctuations. If λ is equal to 0.65 an accurate thermal control can be obtained, instead when λ is equal to 0.5, the original Verlet algorithm is retrieved. The mass of each particle is taken identical and equal to one, meaning that the force is equal to the acceleration.

An example of the application of the Verlet-Algorithm can be found in the harmonic oscillator which can be solved according to the following equations:

$$F = \frac{-dV(\mathbf{x})}{d\mathbf{x}} = -\frac{d}{d\mathbf{x}}\left(\frac{1}{2}k\mathbf{x}^2\right) = -k\mathbf{x}, \tag{4.11}$$

$$a = -\frac{-k}{m}\mathbf{x}, \tag{4.12}$$

By using Verlet algorithm and a forward and backward Taylor expansion:

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \mathbf{v}\Delta t + \frac{1}{2}\mathbf{a}(t)\Delta t^2 + \frac{1}{3!}\dddot{\mathbf{x}}\Delta t^3 + O(\Delta t^4) \tag{4.13}$$

$$\mathbf{x}(t-\Delta t) = \mathbf{x}(t) - \mathbf{v}\Delta t + \frac{1}{2}\mathbf{a}(t)\Delta t^2 - \frac{1}{3!}\dddot{\mathbf{x}}\Delta t^3 + O(\Delta t^4) \tag{4.14}$$

a term by term summation leads to:

$$\mathbf{x}(t+\Delta t) + \mathbf{x}(t-\Delta t) = 2\mathbf{x}(t) + \mathbf{a}(t)\Delta t^2 + O(\Delta t^4), \tag{4.15}$$

which becomes:

$$\mathbf{x}(t+\Delta t) = 2\mathbf{x}(t) - \mathbf{x}(t-\Delta t) - \frac{k}{m}x(t)\Delta t^2 \tag{4.16}$$

that has an error of the order of Δt^4.

4.3 Computational Fluid Dynamics

The governing equations are solved with different methods , such as finite elements and finite volumes methods (Andersson et al., 2011). In particular, in this work the finite volume method (FVM) is presented. FVM is a discretization method to solve partial differential equations. CFD domains are discretized into grid (mesh) composed by elementary volumes and properties can be calculated for each element of the grid. In FVM, the numerical flux is conserved across the cells. Local balances are written for small control volumes, and by using the divergence theorem and integral formulation of the fluxes across the edges of the elements is obtained. Compared to easier methods (such as finite differences), FVM can be used on complex grids, because most of the three-dimensional cases explored by CFD cannot be solved on orthogonal grids (divergence approximated by values along a constant line). Also, the use of FVM ensures that glabl properties are always conserved, because they are built on the concept that properties are already conserved on small volumes. In other words, FVM is inherently conservative.

The computational domain must be divided into a number of smaller flow domains (control volumes or cells). In the center of these volumes, properties are stored, but also boundary points are defined (on the edges of the domain, such as physical boundaries). Once the division is completed and densely populated of elements in areas of interest, transport equations may be integrated over each cell.

The integration process can be explained by the following general rule

$$\frac{\partial v_i \phi}{\partial x_i} = \frac{\partial}{\partial x_i}\left(\gamma \frac{\partial \phi}{\partial x_i}\right) + S_\phi \tag{4.17}$$

where S_ϕ is a generic source term. It is important to highlight that by chosing a set of ϕ, γ, S_ϕ, different governing equations can be obtained. For example, if $\phi = 1$, $\gamma = 0$ and $S_\phi = 0$ continuity equation is obtained. Eq. 4.17 can be integrated over each cell volume V:

$$\int_V \frac{\partial v_i \phi}{\partial x_i} dV = \int_V \frac{\partial}{\partial x_i}\left(\gamma \frac{\partial \phi}{\partial x_i}\right) dV + \int_V S_\phi dV \tag{4.18}$$

At this point, the divergence theorem ensures that a volume integral (i.e. the divergence of convection and diffusion) can be transformed into a surface integral, and the integration domain must be changed from V to δV, the boundary of V:

$$\int_{\delta V} (\mathbf{v}\phi - \gamma \frac{\partial}{\partial x_i}\phi) \cdot \mathbf{n} dS = \int_V S_\phi dV. \tag{4.19}$$

In this way, it is possible to obtain the net fluxes in the volume of control of a generic variable ϕ. The procedure applies for all the elements of the computational domain.

The surface integral can be discretized as follows:

$$\int_{\delta V}(\mathbf{v}\phi - \gamma\frac{\partial}{\partial x_i}\phi)\cdot\mathbf{n}dS \approx \sum_k(\mathbf{v}\phi - \gamma\frac{\partial}{\partial x_i}\phi)_k\cdot(\mathbf{n}dS)_k), \tag{4.20}$$

Fluxes are calculated between the boundary (k) of the cell, while the product $\mathbf{n}dS$ represents the area invested by the flux. This approximation is second order accurate. If two dimensional grids are considered, the following example can be explained:

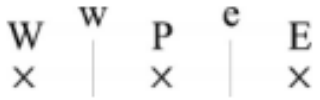


Figure 4.1. *Cells defined by their position (N, S, E, W). The center of the cell is reported in capital letter.*

If we consider the momentum transport between neighbouring cells (i.e. N: North, E: East, W: West, S: South - Centers of the volume of neighbouring cells; n - e -w - s, centers of the faces of neibouring cells), the momentum equation can be integrated over the volume. In what follows, a two-dimensional example is reported:

$$\int\int_\sigma\frac{\partial}{\partial x}(\rho v_x^2)+\frac{\partial}{\partial y}(\rho v_x v_y)dxdy = -\int\int_\sigma\frac{\partial P}{\partial x}dxdy+\int\int_\sigma\frac{\partial}{\partial x}\left(\mu\frac{\partial v_x}{\partial x}\right)+\frac{\partial}{\partial y}(\mu\frac{\partial v_x}{\partial y})dxdy \tag{4.21}$$

that according to the fluxes of neighbouring cells becomes:

$$\left[\int\rho v_x^2 dy\right]_w^e+\left[\int\rho v_x v_y dx\right]_s^n=\left[\int\mu\frac{\partial v_x}{\partial x}dy\right]_w^e+\left[\int\mu\frac{\partial v_x}{\partial y}dy\right]_s^n-\underbrace{\int\int_\sigma\frac{\partial P}{\partial x}dxdy}_{\approx\left(\frac{\partial P}{\partial x}\right)_P\Delta x\Delta y}, \tag{4.22}$$

and the pressure term is evaluated at the center of the cell, eventually interpolated between neighbouring cells. An example of approximation of the convective term is given by:

$$\left[\int\rho v_x^2 dy\right]_w^e+\left[\int\rho v_x v_y dx\right]_s^n\approx\left[\rho v_x^2\Delta y\right]_w^e+\left[\rho v_x v_y\Delta x\right]_s^n \tag{4.23}$$

$$\left[\rho v_x^2\Delta y\right]_w^e+\left[\rho v_x v_y\Delta x\right]_s^n=(\rho U\Delta y)_e\mathbf{v}_{\mathbf{x}e}-(\rho U\Delta y)_w\mathbf{v}_{\mathbf{x}w}+(\rho V\Delta x)_n v_{xn}-(\rho V\Delta x)_s v_{xs} \tag{4.24}$$

where values of the cells are obtained through interpolation. Different schemes can be used to approximate cell face values, such as first-order and secord-order upwind, centered schemes (Andersson et al., 2011)

4.4 Quadrature Method of Moments

Population Balance equation cannot be directly solved in the version formulated in Chapter 3 because of closure problem. However, a possible solution can be obtained by solving the quadrature method of moments (QMOM) (Marchisio and Fox, 2013). This method has been initially developed for mono-dimensional integrals and then extended to more complex cases. In the case of univariate distributions, Gaussian quadrature theory applies. In the Gaussian theory, the moments of the number density function must exist in the integration interval and they are given by:

$$m_k = \int_{\sigma_\xi} n(\xi)\xi^k d\xi, k = 0,1,2..., \tag{4.25}$$

Nodes and weights used to approximate the distributions are used to solve the transport equations of the moments. This is a closure problem that can be synthethized in the calculation of the integral:

$$I = \int_{\sigma_\xi} n(\xi)g(\xi)d\xi) \tag{4.26}$$

where the terms contained in the integral represent the number density function $n(\xi)$ and all the remainin terms grouped together in the function $g(\xi)$, while the integral domain, in the case of the dimension of the particles ranges from $[0, \infty)$. Since the number density function is not known, the numerical scheme is applied by using the transported moments.

Also, orthogonal polynomials must be introduced. These polynomials are respectively orthogonal and orthonormal in the integration interval if:

$$\int_{\sigma_\xi} n(\xi)P_\alpha(\xi)P_\beta(\xi)d\xi = \begin{cases} = 0 & \text{for} \alpha \neq \beta \\ > 0 & \text{for} \alpha = \beta \end{cases} \tag{4.27}$$

$$\int_{\sigma_\xi} n(\eta)P_\alpha(\xi)P_\beta(\xi)d\xi = \begin{cases} = 0 & \text{for} \alpha \neq \beta \\ = 1 & \text{for} \alpha = \beta \end{cases} \tag{4.28}$$

Integration of the weight function over the domain defines a family of polynomials $\{P_\alpha(\xi)\}$, and for any sets, three consecutive polynomials are related by following recursive relationship:

$$P_{\alpha+1}(\xi) = (\xi - a_\alpha)P_\alpha(\xi) - b_\alpha P_{\alpha-1}(\xi), \tag{4.29}$$

and the a_α and b_α coefficients can be calculated according to formulas not reported here for simplicity. By using Eq. 4.29 it is possible to calculate a sequence of orthogonal polynomials to the weight function. The coefficients of the polynomials can be written in terms of moments of the number density function, such that to obtain a polynomial of order N, $2N$-1 moments are needed. For example, if one wants to obtain the polynomial $P_2(\xi)$, the following coefficients can be calculated from the moments of order zero, one and two:

$$a_0 = \frac{m_1}{m_0}, \tag{4.30}$$

$$a_1 = \frac{m_3 m_0^2 + m_1^3 - 2m_2 m_1 m_0}{m_2 m_0 + m_1^2 - 2m_1^2 m_0}, \tag{4.31}$$

$$b_1 = \frac{m_2 m_0 + m_1^2 - 2m_1^2 m_0}{m_0^2}, \tag{4.32}$$

Gaussian quadrature uses the roots of polynomials orthogonal to the number density function as nodes of the following approximation of the integral:

$$\int_{\sigma_\xi} n(\xi) g(\xi) d\xi \approx \sum_{\alpha=1}^{N} w_\alpha g(\xi_\alpha), \tag{4.33}$$

where w_α and ξ_α are the weights and nodes of the quadrature. The degree of accuracy of the Gaussian quadrature is equal to $2N - 1$ where N is the number of nodes, almost two times the accuracy of interpolation using Newton-Cotes to obtain the nodes.

By using the definition of degree of accuracy for a Gaussian quadrature of order N, it is possible to calculate $2N - 1$ moments of the number density function by solving:

$$m_0 = \sum_{\alpha=1}^{N} w_\alpha, m_1 = \sum_{\alpha=1}^{N} w_\alpha \xi_\alpha, ..., m_{2N-1} = \sum_{\alpha=1}^{N} w_\alpha \xi_\alpha^{2N-1} \tag{4.34}$$

which can be solved in different ways, in particular PD used in Fluent User-Defined-Function and Wheeler used in MATLAB, algorithms are reported in Appendix B.

By using the QMOM, the PBE introduced in Chapter 3, in the form of NDF can be re-written by using this quadrature approximation for $\xi = d$:

$$\frac{\partial m_k}{\partial t} + \nabla \cdot (\mathbf{v_i} m_k) = \sum_{a=1}^{N} w_a g(d_a) \Big[\int_0^{+\infty} \beta(d, d_a) d^k \mathrm{d}d - d_a^k \Big], \tag{4.35}$$

where breakage kernel and daughter distribution function can be expressed in different ways.

Daughter distribution function

Depending on the nature of the disperse phase, different distribution functions can be used to describe the number of daughter droplets that originate from a mother droplet. The most general description of daughter distribution function can be obtained by summing delta functions:

$$\beta(\xi_p|\xi_p') = \sum_{i=1}^{v} \delta\left[\xi_p - \xi_i(\xi_p')\right], \tag{4.36}$$

where v is the total number of daughters, ξ_i is an internal function to link the internal coordinates between daughter and parent. Different daughter distribution may be obtained, and in this specific problem, a binary distribution is used, namely from one parent, two symmetrical daughters are created. An example of binary breakage can be reproduced by using the following function:

$$\int_0^{+\infty} \beta(d, d_\alpha) d^k \mathrm{d}d - d_\alpha^k = \frac{3240 d_\alpha^k}{(k+9)(k+12)(k+15)} - d_\alpha^k, \tag{4.37}$$

It is easy to see that when k is equal to zero, the fraction becomes two, meaning that from one single breakage event, two identical droplets are created.

Chapter 5

Computational Details

5.1 Introduction

The computational codes employed in this thesis are described in this chapter. Different codes can be used to describe each scale, in particular LAMMPS, Gromacs, DL_Meso (Seaton et al., 2013), are suitable to describe the molecular scale, while OpenFOAM and Ansys Fluent, can be used to describe the equipment scale. Object-oriented coding languages are extremely versatile in computer simulations. Different objects, similar to big empty boxes, can be filled with different functions. Handling objects can provide a more structured way of solving problems, where so many different models have to be treated. Indeed, it is sufficient to provide a general workflow of operations and call coded objects that contain pieces of information.

Particular attention is given to LAMMPS, Ansys Fluent and OpenFOAM. A short description of these codes is given, together with all the information needed to reproduce the simulations performed in this work.

LAMMPS

LAMMPS (Plimpton, 1995), a free open-source C++ code, originally developed in F77 (Fortran) and F90, is used to simulate phenomena happening at the molecular scale. In general, it integrates Newton's equations of motion for different species (atoms, particles, molecules, beads...) interacting via different forces. The code was written in order to enhance parallel computing, reducing efficiently the computational time as the number of processors is increased. Best performances can be obtained by simulating rectangular boxes with uniformly distributed particles. Scaling and parallel computing are extremely important factors in computer simulations. LAMMPS, as many other codes, includes a Message Passing Interface (MPI) that is able to handle the transfer of data between different cores. Also, its structure allows LAMMPS to be built as a library, hence can be invoked by other codes by using wrappers. Entities that can be handled by LAMMPS are: atoms, coarse-grained particles, proteins, DNA, metals,

granular materials, finite-size spherical and ellipsoidal particles, rigid collections of particles, point dipole particles, and combinations of these. Although is very simple to install and provides good performances, LAMMPS does not have a graphical user interface, build molecular systems, automatically assign coefficients, visualize and output results. The code comes with no warranty and it is distributed under the GNU Public Licence (GPL). This means that any user-defined model can be implemented by directly coding into the source code.

OpenFOAM and Fluent

Continuum modelling has been assessed with two codes for two different purposes. We used Ansys Fluent to perform CFD simulation of emulsions and OpenFOAM to develop a new tool that is able to connect CFD and DPD together into a single code. The two codes can be used alternatively to solve CFD problems. They use the finite volume methods and include different mathematical models to solve the governing equations behind mass, momentum and energy transport. Solutions are given by reproducing average quantities, or fields, that are representative of specific properties of the assessed system. The main difference between the code lies in their commercial/non-commercial nature. OpenFOAM is an open-source code, while Ansys Fluent is a commerical code. Using a non-commercial code requires strong knowledge of programming languages because the models implemented in the code, or ”solvers”, need to be modified or tuned according to the problem that one wants to solve. Also, within OpenFOAM, it is possible to include in-house developed routines and call libraries belonging to other softwares. The power of the open-source feature is limited by a poor graphical interface, which is reduced to text files that are called by the main routine. On the contrary, Fluent is a user-friendly software that allows less skilled users to run simulations. This means that the models already implemented cannot easily modified, even if extra routines can be added. Also, in Fluent many ”limitators” are implemented to avoid simulation divergences. The final worth-mentioning aspect lies in the support. While Fluent has its own technical assistance, OpenFOAM can rely on a huge community of users that share solvers and tips.

5.2 Simulation setup

5.2.1 Coarse-Grained model: LAMMPS

In this part, the main steps to setup a LAMMPS simulation are reported in detail. There are not so many DPD simulations reported in the literature by using LAMMPS, hence, a clear description of a test case is necessary. LAMMPS inputs are read line by line, meaning that it is important to follow a defined path in order to avoid errors. In this specific example, guidelines will be given to simulate a system of DPD particles of

one specie (A) at equilibrium. A test case, *in.water*, can be downloaded in the Section 5.2.5 and in the Appendix A.

Pre-Processing

The set of rules that has to be defined in order to perform a LAMMPS simulation is now described in detail. It is useful to create an identity for parameters that will not change during the simulation and are defined by the user. These are called variables, and can be defined as follows:

Listing 5.1: Code

```
1 variable myname equal 10
```

it defines a variable named myname and puts its value equal to 10. To use this variable in the code, the syntax $*myname* must be adopted. The reference units can be set to lj (Lennard – Jones), for our specific case, since DPD units are non-dimensional and peculiar for this technique, but they are not implemented in LAMMPS. Moreover, lj will guarantee a normalized Boltzmann constant value equal to 1.

Listing 5.2: Code

```
1 units lj
```

Variables and units can be defined and modified at every line of the code, but it is good practice to define all of them during this initial phase. Also, the integration timestep can be defined at this point. The given value will have the units according to the set previously chosen.

Listing 5.3: Code

```
1 timestep 0.01
```

Simulation Domain

Defining a computational domain is essential. A simulation box can be defined in different ways, for example by defining regions and then merging them together. Each region needs geometrical boundaries that can be defined by the initial and final coordinate (units defined by the user via units command line). Simulation boxes can have different shapes (cubes, spheres, ...), and particles interact within these regions. Different regions can also contain different species of particles. If specific geometrical patterns need to be created, the command lattice can be used, otherwise the particles will be positioned in random location across the simulation box.

Listing 5.4: Code

```
1 lattice none 1
2 region myRegionName prism 0 ${xsize} 0 ${ysize}  0 30 0 1 0
```

```
3 create_box 1 myRegionName
```

In these lines, no lattice is given, one single region, originated in (x,y,z) = (0,0,0) and ending in ($xsize$,$ysize$,30), is defined. The shape is a prism and tilting factor can be also defined (0,1,0 in this case). myRegionName is the name provided by the user. Different regions can be defined in this way, merged through a specific command and then converted into a real simulation box using the create_box command.

System settings

In this section, interaction style, boundaries and extra-info can be defined before filling our simulation box with particles. We created an empty region using the previous commands, but in many simulations, it is important to define the behaviour of particles crossing the physical domain. It is quite common to use periodic boundary conditions, as described in the Chapter 3, to assess bulk properties or non-physical boundaries. The kind of interaction between particles can be also defined at this point. Different styles can be selected from the models implemented in LAMMPS, and some of them can be also combined to describe complex systems (e.g. DPD polymers).

Listing 5.5: Code

```
1 atom_style dpd
2 boundary p p p
3 comm_modify vel yes
```

In these lines, interaction style is set to DPD, meaning that DPD forces are acting between particles (beads representing clusters of atoms). Each boundary (x,y,z) is periodic and replaced by a replica of the box, hence, when a particle crosses one edge, a replica of that particle enters from the opposite boundary. Positions of particles must be stored and communicated between processors, but when DPD is used, it is important also to store the velocity of each particle crossing the boundaries, because of the presence of dissipative forces that depend on the relative velocity between beads.

Particle

At this point, an empty region and the style of interactions between beads are defined. Now, it is possible to provide information related to the kind of each bead and fill the simulation box, before running the testcase:

Listing 5.6: Code

```
1 create\_atoms 1 random 3000 123456 NULL
2 mass * 1.0
3 neighbour 1.0 bin
4 neigh_modify delay 0 every 1 check yes
5 pair_style dpd ${T} ${cutoff} 123456
6 pair_coeff 1 1 25.0 4.5
```

create_atoms command will create 3000 particles of type 1, located in random positions within the box, they will fill all the regions, since no specific place is selected ($NULL$). A user-defined seed (e.g. 123456) must be provided to generate random numbers. Each particle in the box (* is a wildcard, meaning all the types), will have mass equal to 1. A list of j-particles that can interact with an i-particle is constructed using a spherical skin (e.g. radius equal to 1.0 length unit) surrounding i. All the particles that are outside this skin, are ignored in the computation of the forces. Neighbouring list can be built at every time step (delay 0 every 1) or after several timesteps, depending on the simulation setup. It is good practice to reconstruct the list at every timestep if strong flow effects act on the system, because closer particles can move away from their original positions in few timesteps. *pair_style* can be used to define the core parameters of DPD interactions, namely temperature and cut-off distance (also a random number must be included). To conclude this session, each atom of type 1 can interact with atoms of type 1 (2,3,... if more species are present) via DPD potential. a_{ij}, and are defined in *pair_coeff*, while σ_{jj} is calculated via fluctuation dissipation theorem.

Calculation

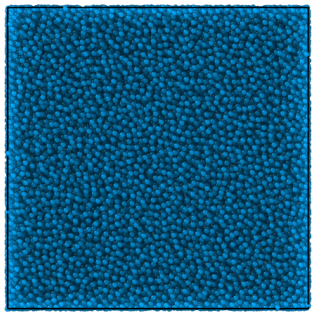
The simulation is ready to be started. Minimization of the energy can be performed on the system before running the integrator. Different integration methods can be selected in LAMMPS, together with *fixes*. Fix command is quite general and contains many different options and operations that can be performed to tune the simulation. Integrators, thermostats, calculations, averages, can be found in *fix*.

Listing 5.7: Code

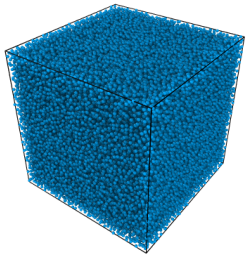
```
thermo 1000
minimize 1e-5 1e-7 1000 10000
fix 1 all nve
fix .........
.........
fix ........
dump .......
run 1000
timestep 0.2
run 1000
```

thermo can be used to define the number of timesteps before printing an output line on the screen. The type of integrator that is chosen for this simulation is NVE, hence, the total number of particles, the volume of the simulation region and the internal energy of the system are conserved, while properties such as pressure, are computed by the code. A dump command can be used to generate videos and pictures of the system at different timesteps. Run starts the simulation, after 1000 timesteps, its size is changed to 0.2 and further 1000 timesteps are computed. Post-processing can be performed in

different ways. VMD can be used to see the dumped images and videos, while all the relevant values can be read on the simulation log that is produced and saved during the run. A picture of the system can be observed in Figure 5.1, where it is clearly showed that the molecular identity is lost using DPD coarse-graining:



(a) *front view*



(b) *3D view*

Figure 5.1. *Dissipative Particle Dynamics representation of a simulation box containing water beads. Each bead (spherical particle) represents a cluster of molecules of water. In this rapresentation, the molecular identity is completely lost and replaced by soft potentials.*

5.2.2 Equilibrium Simulations: Pluronics in Water

Equlibrium and non-equilibrium simulations require a slighlty different sequence of command lines. These extra commands and functions can be found in the tutorial section *(in.pluronic)*. One substantial change regards the description of the interacting species. In fact, a *molecule* file must be provided to describe the bonded interactions between particles belonging to one single polymeric chain. Most the previous steps are repeated or slightly modified. Regarding the DPD parameters used to simulate the polymer chains, they were obtained from the literature and compared against experimental data.

Equilibrium simulations are needed to forecast phase diagrams that can be used to understand the microstructures present when the recipe of a mixture is altered.

Differences in microstructures that can be appreciated only at quasi-molecular level of resolution, can cause differences in physical and transport properties. The way in which phase diagrams can be simulated, using computational tools, consists in equilibrium simulations. In this specific type of simulations, randomly positioned beads, move and explore possible configurations, according to specific integrators, until a minimum of energy is reached. The time in which these structures are formed is called relaxation time. Microstructures usually relax in timescales that cannot be accessed by traditional molecular dynamics. Here comes the necessity of introducing coarse-graining procedure. By varying the number of beads, representing clusters of atoms, it is possible to move along the concentration axis, while the temperature is kept constant during the whole simulation. This work is focused on the reproduction of three copolymers in water, namely Pluronics L64, P104 and P85.

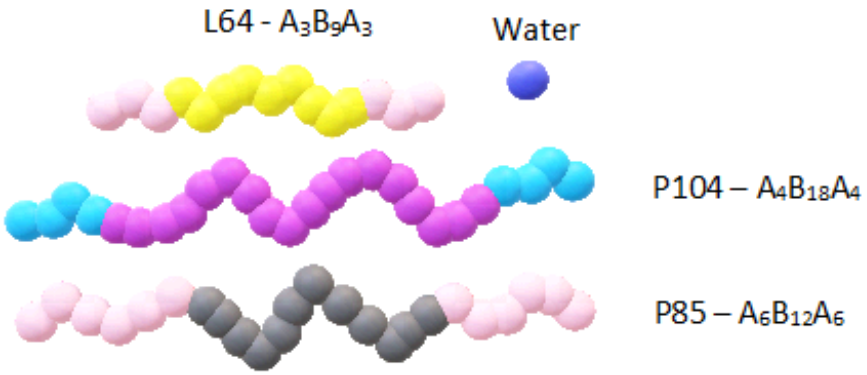


Figure 5.2. *Coarse grained models of the Pluronics L64, P104 and P85 according the described level of coarse-graining.*

The level of coarse-graining adopted to describe the Pluronics chains is 4.3 for the EO repeated units and 3.3 for the PO repeated units. This means that one coarse-grained bead of EO contains 4.3 atomistic EO monomers and the same conversion procedure applies for PO. For example, using this set of parameters, Pluronics L64 chains are composed by 15 beads and simulated as $A_3B_9A_3$ DPD chains, Pluronics P104 is simulated as $A_4B_{18}A_4$, and P85 as $A_6B_{12}A_6$ where A is the coarse-grained bead for the EO unit and B is the coarse-grained bead for the PO one. Simulations of different concentrations of Pluronics in water were performed by varying the number of beads of the two components (i.e. water and Pluronics), keeping the total number of beads in each box fixed (e.g. for a system composed by 81000 beads, if 50% is composed by water, 40500 spherical beads are water-type). Bonded and non-bonded interactions between beads are accounted for in the DPD model (Prhashanna et al., 2016). The former was described using both harmonic and FENE potentials, while the latter are reported in Table 5.1. All values are reported in DPD units.

Table 5.1: Interaction parameters for the three species, Water, PEO and PPO. All values are reported in DPD units.

	Water	**PEO**	**PPO**
Water	25.00	54.00	66.00
PEO	54.00	25.00	34.00
PPO	66.00	34.00	25.00

The dissipative parameter γ was set equal to 4.5 (in DPD units) for all the species, while the stochastic parameter σ was set equal to 3 according to the fluctuation-dissipation theorem, when the value of k_BT is equal to 1. Simulation boxes of different sizes were tested, from $20 \times 20 \times 20$ cut-off radii to $40 \times 40 \times 40$ cut-off radii. The simulation box of $30 \times 30 \times 30$ cut-off radii was found to be a reasonable compromise between reduction of simulation box artifacts and acceptable simulation times. The initial configuration of the system is prepared by random positioning of water beads and Pluronics L64 chains. The number density (i.e. number of beads per unit volume) was set equal to 3 DPD units, meaning that the total number of beads was 81000.

The cut-off radius for non-bonded interactions was set equal to 1.00 with a timestep of 0.01 DPD units. Equilibrium simulations were carried out for 2×10^6 timesteps, while non-equilibrium simulations were carried out for 3×10^5 followed by 5×10^5 timesteps applying different shear rates in different simulations. The range of concentrations spans from 5% to 95% in weight percentage (w/w) of Pluronics and the range of non-dimensional DPD shear rates varies from 0.005 to 2. The DPD energy was set equal to 1 and its value was recorded every 500 timesteps. Moreover, the DPD property of preserving hydrodynamic interactions ensures that momentum is conserved across the box. The velocity Verlet algorithm was used as integration scheme. During the overall simulation time, energy was stable at $1.0 \pm 0.01\ k_BT$.

5.2.3 Non-Equilibrium Simulations: Lees-Edwards Boundary Conditions

A peculiar aspect of DPD is the conservation of momentum. All the species are simulated together with the solvent in an explicit way, such that all the possible interactions are computed. The concept of momentum being conserved perfectly matches with the Lees-Edwards (LEBC) model to simulate flowing effects (Chatterjee, 2007; Fedosov, Karniadakis, et al., 2010). The original model is based on a simple concept: to reproduce a fluid under extreme shear conditions, it is possible to replace the whole system with a small sample of it. The simulation box contains particles that undergo a velocity gradient in the vertical direction. The simulation box, A, is surrounded by replicas of itself, B and C, but while A is fixed, the surrounding boxes (B and C) slide in opposite directions, with a specific speed. A linear velocity profile

can be obtained in this way, and the value of the velocity becomes zero in the middle of the box. Periodic boundary conditions must be adapted to this model, because of the streaming (sliding) effect that add an extra path when a particle crosses the boundary as it proved in Fig. 5.3 (Lees and Edwards, 1972).

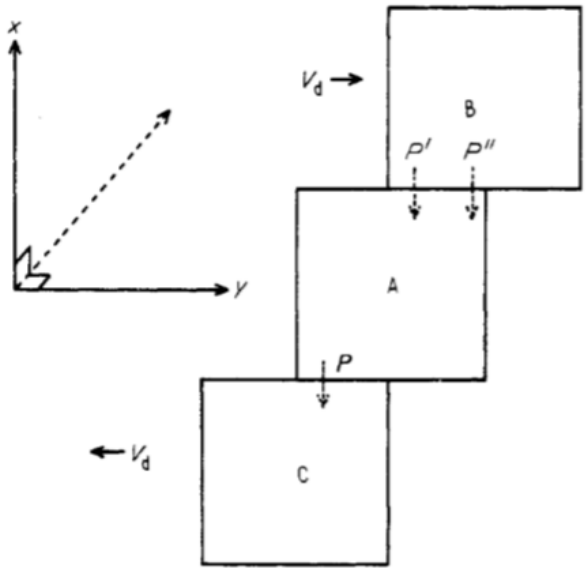


Figure 5.3. *Lees-Edward boundary conditions are explained. The box A is main simulation box and it contains P particles. On the top and bottom, replicas of the box move at the same velocity but in the opposite direction, causing the development of a linear velocity profile across the simulation box A. Modification of the periodic boundary condition can be also appreciated. When P leaves the box, its replica does not re-enter as a P' but as P" because of the sliding velocity on the top of the box. In fact, the new position must consider the distance covered by the particle during the new timestep due to the velocity at the top of the box.*

When a particle, P, leaves the simulation box and migrates into a replicas crossing the boundaries, a clone of P should be reintroduced in P'. However, the velocity of the particle P, which left the box from the bottom part, is different from the velocity of the particles that are in the top and this drift velocity must be summed to the original P velocity. This results in a shift in the position from P' to P''.

This model is not directly implemented in $LAMMPS$. However, it is possible to reproduce $LEBC$ via fixes. Different values of shear stress can be obtained through the application of different velocities on the beads that are close to the boundaries (top and bottom) of the simulation box. The maximum value of the velocity at the top of the box is equal to $\dot{\gamma} l$, where $\dot{\gamma}$ is the shear value imposed on the system and l is the length of the box. If the conservation of momentum is respected, a linear velocity profile, across the simulation box, is obtained. The magnitude of the shear stress should

generate velocities that are larger than the thermal velocity of the beads, leading to meaningfully observable shear flows in computational studies. The way in which LEBCs are reproduced into LAMMPS follows:

Listing 5.8: Code

```
velocity all ramp vx 0 ${velramp} y 0 ${ysize}
fix shear all deform  1 xy erate ${srate} remap v flip yes unit box
```

The system is initialized with a linear velocity profile (*ramp*), applied to all the species (*all*) and modifying the x-component of the velocity, $\mathbf{v_x}$, from 0 to a user-defined value, along the y direction. *fix deform* can be used to reproduce the streaming effect by imposing the *erate*, engineering rate, that applies a constant shear on the simulation box. This command can have many different inputs such as the velocity of the walls, the strain rate, oscillation period and so on together with the possibility of ipping the box if it becomes too skewed.

A triclinic box is used to perform non-equilibrium simulation. Triclinic boxes need tilt factors to be defined. Tilt represents a displacement which is applied to a orthogonal box in order to become a parallelepiped. They are expressed as xy xz and yz in LAMMPS and represent a limit in the skewing of the box (no more than half a distance on the parallel direction, e.g xy - x direction is the parallel). The approach used so far consists in using an "constant engineering shear strain rate" (erate) applied on the xy tilt factor (volume is constant and such that the box is not deformed only in one direction). The erate changes the tilt factor (i.e. displacement of the box in parallel direction) at a constant rate and its units are $1/time$. Shear strain must be unitless and it usually takes the dimension of an o set/length (perpendicular to shear direction). An example of tilt equivalent boxes is given. If x length is 10, the tilt distance is ± 5, thus all the tilt configuration -15, -5, 5, 15, 25 are equivalent. For higher shear rates, a ipping option is possible in order to avoid skewness problems. Periodic BC must be used in at least the parallel direction, otherwise atoms positions are not remapped. Remap is possible either in position (adjusting the position for an atom which migrates to a replica) and velocity (a delta of velocity is added to the crossing atoms).

5.2.4 Non-Equilibrium Simulations: Pluronics in Water

In non-equilibrium simulations, each system was initialized with a linear velocity profile, with the maximum desired velocity at the top of the box and zero velocity at the bottom. Shear was only applied on the xz plane, meaning that only P_{xy}, one of the three non-diagonal components of the stress tensor, was not null. The velocity on the top slab was set equal to $\dot{\gamma} l$, where γ is the DPD shear rate value and l is the length of the box. In our range of investigation (i.e. from 0.005 to 2 DPD shear rate), a linear velocity profile was obtained for both a system containing only water and a mixture of water and Pluronics L64.

In order to ensure the validity of the results in the operative range, two sets of tests were performed for the upper and lower limits of the shear range. To set the upper limit, we observed the behavior of water viscosity, which needs to be consistent with its Newtonian nature. However, for shear rate values greater than 2 DPD units an unphysical dependence of the viscosity on the shear rate is obtained. A shear rate of 1 DPD unit is therefore the maximum applicable in our simulation set up.

It must be highlighted that, the set of parameters used to describe the Pluronics L64 chains is valid in equilibrium conditions and non-equilibrium parametrization may differ such that the predictions of some equilibrium properties could results in unrealistic values. The shear rate in DPD units can be different from the physical shear rate at which an analogous situation is reached. When the value of the shear is greater than 1 DPD unit, the system could be exposed to extreme deformation that lead to non-physical results. To set the lower operating limit, velocity profiles across the simulation box at different shear rates were analyzed. When the shear value imposed on the system is around 10^{-3} DPD units, the thermal fluctuations due to the DPD thermostat are masking shear effects and the velocity profile is affected by beads, moving according to the temperature of the system. This effect was tested on both water and water-Pluronics L64 mixtures, therefore shear rate values smaller than 10^{-3} DPD units cannot be explored. Concluding, by using a conservative approach we can set the operating range of shear rates between 0.005 and 1.

Viscosity was obtained by averaging the value of the non zero component of the stress tensor, P_{xy}, every 100 timesteps. All the viscosity values are recorded after an initial equilibration phase, such that initial fluctuations are filtered. The final value was recorded when fluctuations were around ± 0.01 by adjusting the simulation time window. In particular, the trend of the viscosity was recorded during the simulation time and the final value recorded only when fluctuations were in the order of magnitude of 0.01.

The harmonic potential was finely tuned in order to suppress the formation of over-elongated chains, hence the coefficient was initially set equal to 4.0 (in DPD units) and then modified. One concentration (i.e. 25% w/w of Pluronics L64 in water) was used as a sample and κ_{Harm} was increased until variations in the viscosity were negligible. In order to reduce potential errors due to extreme shear conditions and over-elongation, the same set of parameters used for the harmonic potential was used in the FENE potential simulations, this means that the value of the spring constant, was set equal to 50 DPD units, while the equilibrium distance, r_e, was set equal to 1.00 DPD units. Rheograms were obtained for different concentrations of Pluronics L64 at different shear rates, recording the value of viscosity every 0.01 DPD shear units. The qualitative variation of the trend of the viscosity was proven to be related to differences in the microstructure.

5.2.5 Code Download

A simulation setup file can be downloaded at:

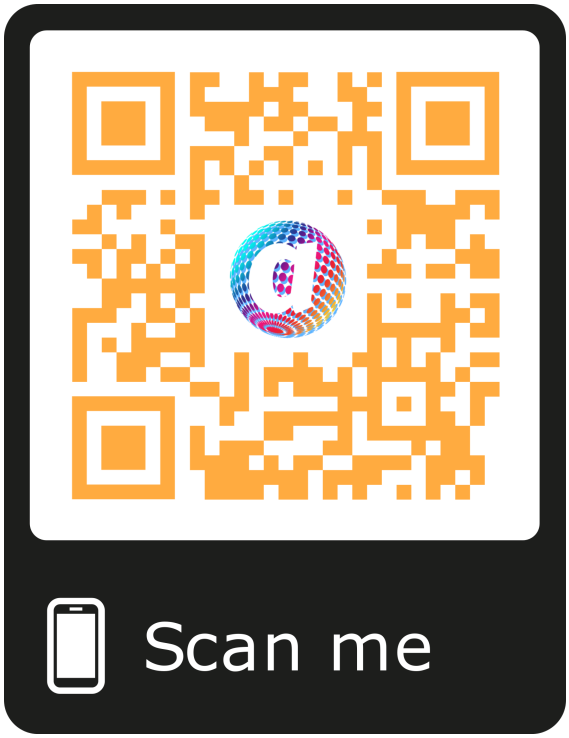


Figure 5.4. *QR code can be used to download Lammps simulation files.*

The input scripts, in.water, in.pluronic and in.shear, can be found in /test/, also some tutorials and manuals are provided in the parent folder. Instructions about how to run a simulation are reported in the README file. Also see Appendix A for an example of Lammps simulation setup file.

5.3 Clustering Algorithm

A cluster algorithm was introduced in this work, in order to identify and characterize the shape of the microstructures in equilibrium and non equilibrium simulations. The cluster algorithm was in-house developed and coded using Python (Python Software Foundation. Python Language Reference, version 2.7. Available at http://www.python.org). Three main parts were written for the purpose: a general code.py, where the main workflow and operations are executed, a fnc.py, where all the functions were stored, and a gui.py, that could be used to enable a small graphical interface.

code.py

The code takes as input a trajectory file (*xyz* output file from LAMMPS), where all the coordinates of different particles are stored for each timestep. Coordinates and the current timestep are stored into a Python object called *times*. Each *time* contains a sub-structure, *TimeStep*, where the coordinates x, y, and z are stored together with extra-values (e.g. an id number). Three different flags can be activated, *hist*, *ave*, and *pbc*. The first flag generates histograms representing the distribution of the clusters at each timestep, the second flag averages clusters and histograms during a certain amount of timesteps, and the last flag is used to include periodic boundary conditions in cluster identification.

The initial part of the code is used to read and store data. The file named "*namefile.xyz*" is scanned and each line that does not contain a trajectory information is excluded. The number of beads, hence the lines contained in each timestep, is known a-priori, and a counter is increased after every accepted line. Once the loop all over the beads is finished, the position *i* of the structure *times[i]* is updated. This operation is repeated until the file is over, and then closed. At this point, the memory contains all the coordinates of each bead at every single timestep. A *frameskip* value can be selected before starting the second loop, meaning that not all the simulation timesteps will be used for computing averaging and for the graphical outputs.

The second loop also requires an initial and a final timestep on which the cluster algorithm run. Inside this second operation, for every single timestep which is not skipped, all the columns contained in the list *times*, are separated and individual lists of x-coordinates, y-coordinates and z-coordinates are obtained. Here, the *pbc* flag discerns between two different options. If *pbc* is set equal to 1, the clustering operation is not performed by using the coordinates of each particles but by calculating the relative distance between all the particles, hence a *distance_matrix*, is used instead of the one containing the positions.

One problem of this specific case is represented by the *distance_matrix*, which can be extremely computational expensive, hence parallelising the calculation could be necessary. If the *pbc* flag is set equal to 0, a density based algorithm (Ester et al.,

1996), described in the next sub-section, takes all the coordinates contained in one timestep and compute the number of different clusters. Each cluster is identified by a different colour and the number of particles having the same colour is stored at each calculation. Two important aspects have to be checked at this point, the value of the minimum distance to distinguish one cluster from another, ϵ_{py} , and the total mass balance. One line, recording the mass balance, namely the number of particles found by the algorithm against the number of initial beads in the system, is outputted at every clustering operation. Tuning on the ϵ_{py} parameter might be necessary to avoid losing beads. In this work, we used a value ϵ_{py} equal to 2.0, and we verified that all the beads are found at each timestep by the cluster algorithm. This ensures the conservation of the mass into our simulation box.

In the second loop, three different outputs are obtained. The first is the number of clusters against the simulation time. For each analyzed timestep, the number of total clusters identified are stored and plotted against the simulation time, such that it is possible to appreciate how the system evolves in time by modification of the microstructures. If the *histo* flag is set equal to 1, the instantaneous cluster mass distribution is plotted. In one single timestep, many clusters with different size can be identified in this way. Also, snapshots where different colours identify different structures (which in this case means stand-alone, non-connected aggregates) are produced at this stage. If the system is composed by spherical or elongated micelles, the gyration radius is also computed according to the following equation:

$$R_g = \sqrt{\frac{k_{py} \sum_n \left((x_i - x_{CM})^2 + (y_i - y_{CM})^2 + (z_i - z_{CM})^2 \right)}{n}} \tag{5.1}$$

where k is equal to 3/5, x_{CM}, y_{CM}, z_{CM} are the oordinates of the centre of mass for each single cluster, and n is the number of beads contained in one cluster. The code file ends with the averaging functions that are described in the func.py header. Such functions are used to track the evolution of the cluster mass distribution over a certain number of timesteps and can be repeated depending on the time windows that one wants to investigate.

fnc.py

In this header, all the functions and objects are coded, implemented and called by the main *code.py* script. One object is identified as a container of lists, *TimeStep*, where an id, x, y, z coordinates, histogram value for beads and corrected histogram value for the polymeric chains are stored. After its definition, an initialization procedure (constructor rule) must be given. Following this part, six main functions were written to pre-process and post-process data:

- *add_atom*: this function takes as an input an id value and the x, y, z coordinates and appends all these coordinates to the lists (xCoord, yCoord, and zCoord). The

output of this function is to create lists within an object to store coordinates for one single line of the data file.

- *AverageManager*: this function takes as inputs the list containing all the cluster sizes, the number of clusters and a storage matrix. This function counts the number of times that an identity, namely one cluster composed by N beads, is found. The output is an updated matrix, containing size of clusters and number of times that they are identified on that single timestep, or also added to the previous evidences if a storage matrix is passed as an input.

- *sortingHistograms* and *rebuildingHistograms*: these two functions are simply used to improve the graphical output of the final histograms. They take as an input the lists of clusters and the number of times each cluster is repeated and, in the first case, put them in crescent order (based on their size), while on the second case, they normalize the number of elements, in such a way that it is possible to have bins of different sizes. The output of these functions is again a modified list.

- *pbc_distance_matrix*: this function takes as an input all the coordinates of one timestep for all the beads and it calculates the distance between every $i - specie$ with all the $j - species$. If the distance is greater than a certain value, a "replica" of the particle is mirrored across the boundary of the box. The output of this function is a matrix containing all the distances between all the particles for one timestep.

- *average_histograms_in_time*; this function takes as inputs the object containing both the lists of coordinates and the number of clusters identified at every single timestep, the starting and ending averaging time, a *frameskip* if one wants to avoid taking all the single timesteps and a *frame_collection* parameter, which depends on the LAMMPS output file (after how many DPD timesteps, the xyz file is written). This function outputs the final plot, where the histogram representing the frequency of finding a cluster containing a certain number of particles in the time windows selected.

5.3.1 DBSCAN Algorithm and quantitative comparison

An implementation of density based algorithm, named DBSCAN (Ester et al., 1996), is the core of the script. Different techniques can be used to identify cluster and aggregates as reported in Figure 5.5:

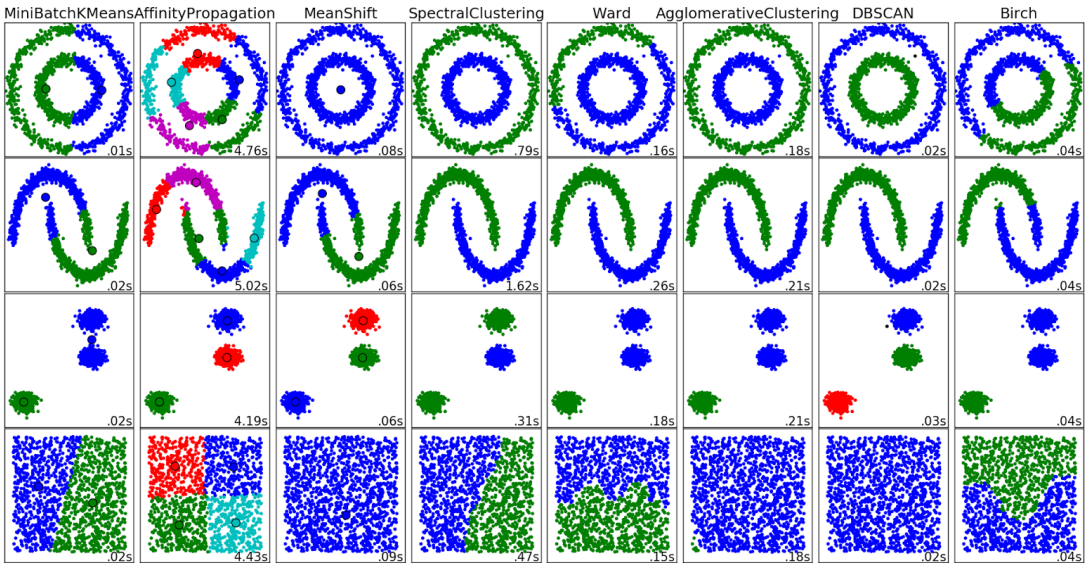


Figure 5.5. *From left to right, column represent clustering algorithm that can be implemented into python. In each column can be appreciated the performances of the clustering algorithm on clouds of points. Different colours represent different clusters identified by the algorithm. In particular, the seventh column represents DBSCAN, which can be used to identify structures such as micelles or distinct clouds of points, in the shortest time and not knowing the number of clusters a-priori.*

DBSCAN is a density-based algorithm for applications with noise. It was proposed by Ester et al. for data clustering, and today is used in Machine Learning and Neural nets. The main concept behind this algorithm is that given a certain cloud of points, it recognizes those who are packed together from those who are "boundary" points (located in low-density regions or at the edge of a cluster). Three different classes of points (or particles or data) can be identified: *core*, which are surrounded by a pre-defined number of reachable neighbors, *reachable*, that are located within a certain distance from the core points and *non-reachable*, which stand on outside layers.

A sphere of radius ϵ is calculated around each point. Spheres containing a user defined number of particles (including the particle itself), are marked as core-points. One cluster is composed by core points, and each core point can be reached from core point belonging to the same cluster. A reachable point, instead can be reached only by one core point, but, in its surrounding, there are less than the user-defined number of particles to be considered as a core. Non-reachable points cannot be reached by any

other point and are isolated.

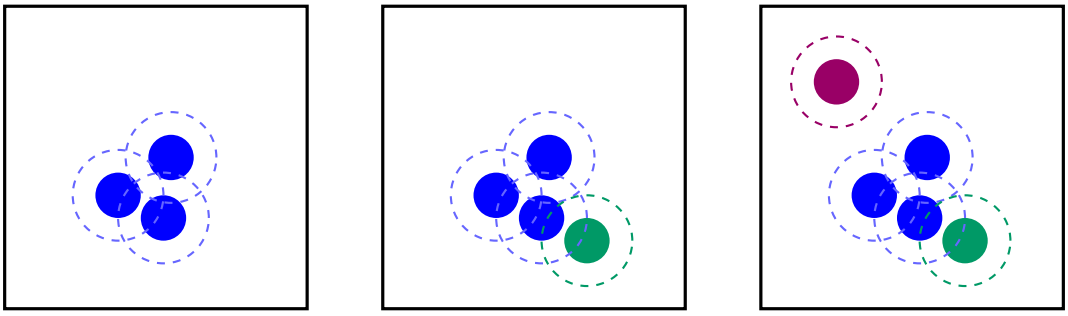


Figure 5.6. *From left to right, three different scenarios where beads are identified as "reachable" if they are closer than a certain distance, hence grouped into the core of one cluster (blue), as boundary points (green) if they are within the distance but they can be reached only from less beads, hence belongin to the same cluster, and "non-reachable" (red) if they are isolated from the remaining beads, hence not counted as belonging to the cluster.*

The purpose of using DBSCAN lies in the high number of advantages obtained with this technique: the number of expected clusters or structures is not necessary known a-priori, meaning that clusters are effectively identified based on the distance between structures and not forced to be clusters (i.e. a minimum/maximum number of clusters is given a-priori); it is possible to recognize arbitrary shapes, even irregular or disordered shapes can be identified by this algorithm; noise can be easily handled and filtered; it is insensitive to the order of the points contained in the database, hence the points must not be ordered before being scanned. The main disadvantage is related to the choice of the ϵ_{py} and the minimum number of points to define a core. These two parameters must be carefully tuned and, in cases where the number of confined particles is extremely high, this cluster algorithm may fail, and assign all the points to a unique cluster (i.e. all the points are core points).

In order to quantify the *sphericity* of the aggregates marked by DBSCAN, the data collected during the cluster analysis was employed to calculate the micelles gyration radius and was used to determine their *sphericity*. It must be highlighted that, for complex structures, there is a correlation between the aggregation number, namely the number of particles inside one cluster, and the gyration radius:

$$N = C R_g^d \tag{5.2}$$

where N is the already introduced aggregation number, C is a constant, R_g is the gyration radius of the micelle and d is a scaling exponent, that tends to three in the case of spherical structures and tends to two in the case of cylindrical or worm-like structures. Plotting the aggregation number versus the radius of gyration (or vice versa) in a log-log scale allows to identify the value of the exponent d.

5.3.2 Code Download

The full version of the code can be download at:

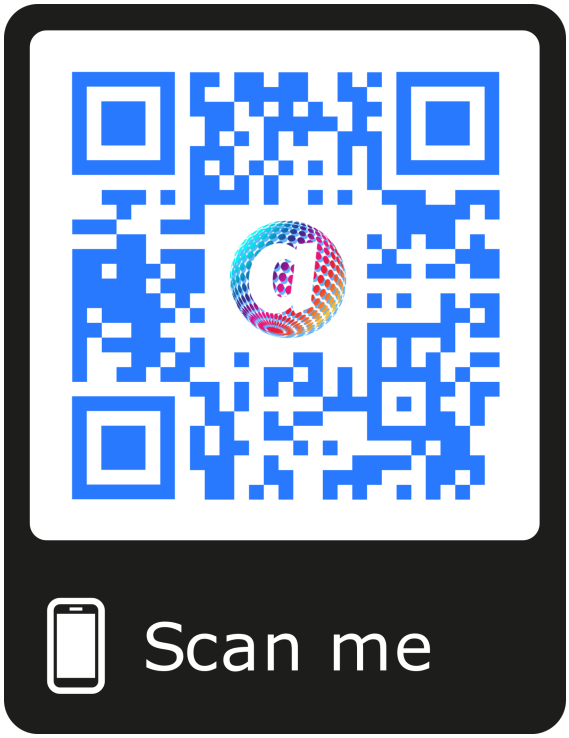


Figure 5.7. *QR code can be used to download the python version of the clustering algorithm*

The folder contains all the files described above, one testcase and a README file with the instructions to run the code. Also see Appendix A for the complete code.

5.4 Continuum CFD Models: Fluent

Manufacturing devices to obtain complex fluids can be also simulated at the equipment scale by CFD. The way in which we assessed the simulation of these systems, was based on focusing on one property of the mixtures that can be measured and at the same time modelled at a macroscale. By using the commercial code Ansys Fluent, we simulated two complex geometries and validated power numbers and droplet size distribution. ESCO 6L and Silverson mixer have been validated under different conditions. Before doing that, we had to implement our own version of the QMOM in Fluent, via User Defined Function (UDF).

5.5 Fluent UDF

A user defined function was used to implemenent and solve the PBE using the QMOM and the kernel described in Chapter 3 and 4. UDF can be used to introduce lines of code into Fluent such that the models already present can be modified and improved. This was done because the implementation of the breakage kernels into Fluent is not sufficient to reproduce our experimental setup. Our UDF used to implement and solve QMOM in Fluent can be found in Appendix A. QMOM six moments of the droplet size distribution (DSD), from zero to five, and it works within the two-fluid model of Fluent. The six moments allow to reconstruct a quadrature approximation with three nodes and weights that is in turn used to overcome the closure problem. This means that one single node of the quadrature is a ”class” of droplet with a specific size. Together with the governing equation of the flow, six more equations related to the transport of the moments must be solved. The way in which nodes and weights are calculated from the moments is the Product-Difference algorithm, already introduced in Chapter 4 and reported in Appendix B. Two kernels have been coded to model the breakage of silicone-oil droplet in water. In particular Alopaeus - Laakkonen and Coulaloglou-Tavlarides kernels were tested.

In order to run a simulation, an initial value for all the moments of the DSD must be calculated. The initial shape of the distribution can be assumed to be log-normal, and the correlation between the moment of order three and the volume fraction of the disperse phase can be used to obtain the moment of order zero, representative of the number of spherical aggregates in the initial disperse phase.

In the UDF, six scalars are used (moments from zero to five) and transport equations are solved for all of them, including source terms. Also a total of thirty-four memories (UDM) was used to monitor and check the moments during the simulation, together with storing variables. In particular, memory UDM18 contains the mean value of the Sauter diameter.

If the volume fraction of the disperse phase is small and does not affect the viscosity of the mixture, such that the velocity field is not altered, it is strongly suggested to

decouple the solution of the flow field from the solution of the population balance equation.

When the velocity field of the two phases is fully developed, UDF can be compiled and loaded. For this purpose it is necessary to have C compiler already installed on Windows. To compile and load the UDF, type on the Fluent TUI: ”define/u-d/functions/compile” and ”compile”. At this stage, it is important that the file.c is present in the working folder. Type a name and wait for the function to be compile. A new folder, named as the C file name, should have been created in the working directory.

Load the function by repeating ”define/u-d/functions/compile” and ”load”. Select the top menu UDF, and indicates the number of memories that have been used. In particular we used thirty-four UDM, and six UDS. For each of the UDS, define that they must be only calculated in the disperse phase, and for ”flux function” click ”mass flow rate” in the drop down menu.

In the menu ”define/materials” or in the materials icon in the left side menu of the GUI, edit the material properties of the secondary phase, by introducing UDS Diffusivity equal to 0.001 for all the UDS.

Now that the library is loaded, it is possible to change the diameter of the disperse phase, in the ”properties” section of the ”phase” section in the GUI, from constant to ”User-Defined Functions” and ”bubble_diameter:..”

Moments need to be initialized and the correct values must be inserted in the boundary conditions. If an inlet is present, it is possible to initialize all the UDS with the correct value of the moments. If there is no inlet, the system must be patched, such that the value of the moments is initialized to a uniform value for the disperse phase.

The last step before running a simulation is to ”hook” the subroutine adjust. In UDF menu, click ”hook” and in the box ”adjust” click ”qmom_adj::...”. Now, select for all the zones present in the simulation, where PBE must be solved, the source terms for all the UDS. In ”define/Cell Zone Conditions” or ”Cell zone conditions” in the left side of the GUI click on the ”fluid” zones and in the window menu, click on the label UDS. For each of the UDS, define the corresponding ”p_source_mX::..” (X from 0 to 5).

Convergence criteria must be quite strict. It is suggested to select at least values of the residuals for the moments equal to 10^{-5} - 10^{-6}. Also, it is suggested to reduce the Under-Relaxation Factors up to 0.01 when oscillating residuals are monitored.

5.5.1 ESCO 6L

Computational Domain

The mixing tank, ESCO 6L (EL-Hamouz et al., 2009), is composed by a sawtooth impeller and a rotating anchor. The operating volume is six litres, the impeller is inclined by a specific angle, while the anchor acts as baffles on the system. The geometrical details are reported in Tab. 5.2.

Table 5.2: Geometrical details of the mixing vessel and operating conditions

Name	Value
Tank Diameter, cm	20.00
Tank Height, cm	30.00
Impeller Type	Sawtooth
Impeller Diamter, cm	5.00
Operating Volume, l	6.00
Rotation Speed, RPM	3000

In Figs. 5.8 and 5.9 are reported the real system and all the 3D CAD models of the mixing tank, impeller and anchor:

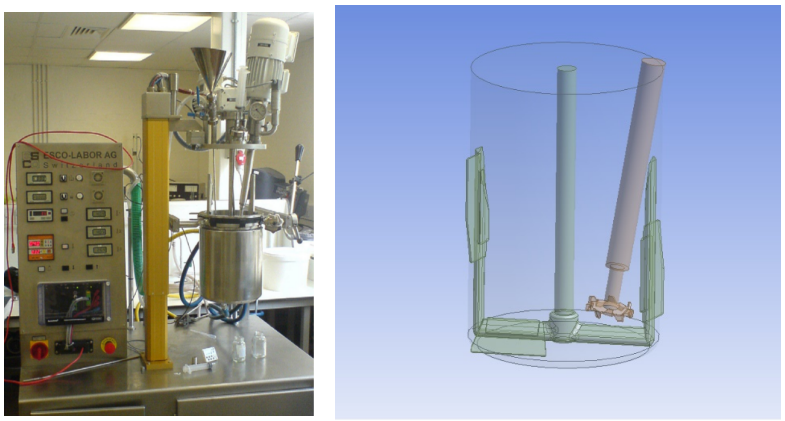


Figure 5.8. *On the left: laboratory scale of the ESCO Mixer in Manchester University. The mixer has a working capacity of 10 litres, an inclined sawtooth impeller and a rotating anchor with scrapers. On the right: 3D CAD model of the ESCO Mixer reproduced with Design Modeller, Ansys. The 3D model has the same dimension of the lab scale one (EL-Hamouz et al., 2009).*

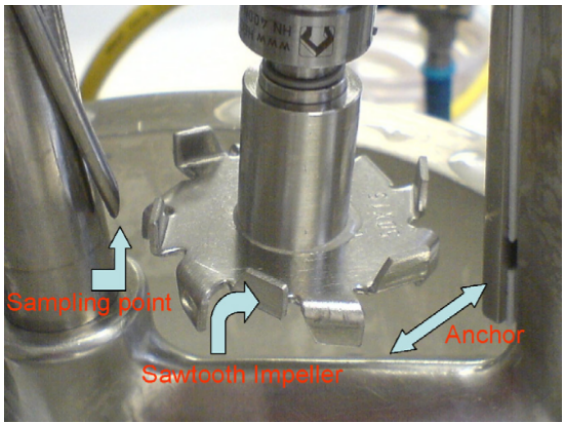


Figure 5.9. *Detail of the sawtooth impeller of the ESCO mixer. Also, the sampling point and the body of the anchor can be identified in the picture (EL-Hamouz et al., 2009).*

In order to simulate the system, different meshes have been tested, from tetrahedrons to hexahedrons dominant. Also, different levels of mesh resolution were tested in order to obtain grid independence, based on the power number obtained. An example of hexahedral mesh is reported below in Figure 5.10:

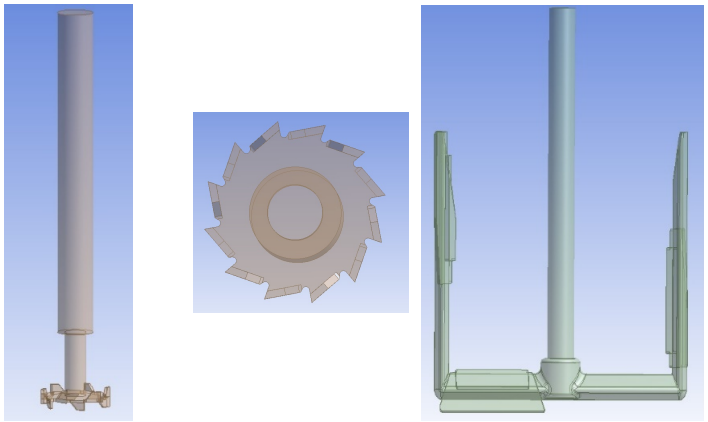


Figure 5.10. *3D CAD detail of the sawtooth impeller and rotating anchor of the ESCO mixer obtained in Design Modelles, Ansys.*

The coarsest mesh is composed by 800,000 cells, hexahedrons dominant, with particular level of detail around the moving objects. Finer meshes containing up to 4,000,000 elements have been also tested and result reported in Chapter 6. Also, anchor was removed in some cases to reduce the computational time, without influencing the results. This approximation could be accepted because during the experiments the anchor was first turned on to improve the mixing between the components and then turned off in offset position, while impeller was rotating. In order to reproduce the motion of the impeller, we created a rotating region around the blades. The solid

part of the impeller and the shaft are static, and no motion is imposed on them. This method is called Multi Reference Frame (MRF) and it is opposed to the sliding mesh technique. In this framework, a rotating region of fluid is created around the impeller and, while the rest of the tank is kept in a steady condition, only the cells belonging to this volume are rotating around the impeller. The mesh is stationary and does not change or slide during the whole simulation. The use of the MRF as already proven in many similar works, is able to provide reliable results in less computational time. Even if the geometries are frozen in space, the volume of fluid surrounding the impeller has a velocity that is able to create the areas of turbulence needed for the modelling of the system by using Population balance equation and breakage kernels. Clearly, a sliding mesh could provide a better description of the system, however due to the high complexity of the geometries, discontinuities caused by the matching between sliding cells, could be cause crashes in the simulation. In our work, we focused on improving the solution in the boundaries between the two references in order to avoid flow field discontinuities that may false the results. Moreover, once the flow field is fully developed, the system could be considered almost steady state, since no external factors contribute to its perturbation. In this case, we have two specific reference frames, the main reference that contains the tank and the anchor, located in the origin of the axis, and the rotating frame, which has origin in the centre of the impeller and the axis is inclined by 8 degrees with respect to the vertical axis.

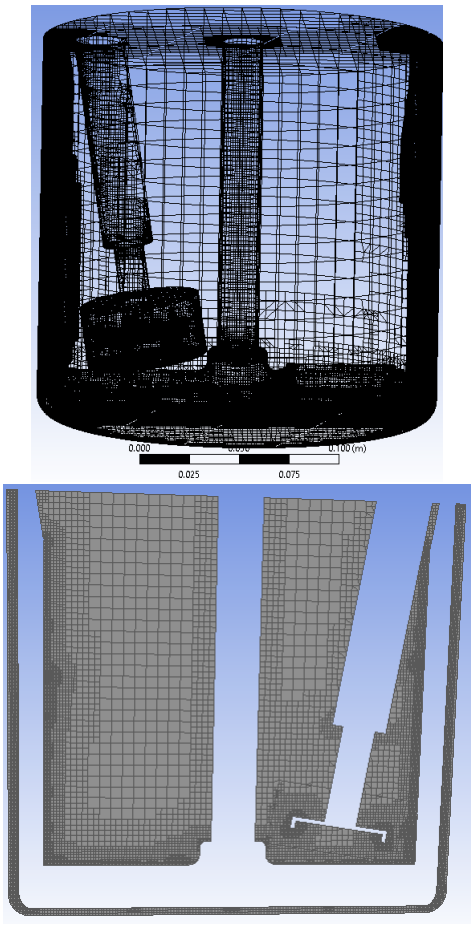


Figure 5.11. *On the top: Example of mesh used in the CFD simulations. The mesh was obtained with the cutcell method and it is composed by almost 800 000 hexahedrons. On the bottom: section view of the mesh, where it is possible to appreciate the level of refinement closer to the impeller region (rotating zone)*

Computational Details

In order to perform a full two-fluid simulation, we started by developing the velocity field, slowly increasing the rotating speed of the impeller. The target for the impeller speed was 3000 RPM, in fully turbulent regime. We started our simulation from a laminar case at 100 RPM and increased after convergence was reached. We tested the system at 100, 200, 400, 500, 1000, 2000 and 3000 RPM. At 1000 RPM, we introduced different turbulence models and tested the performances or Reynolds Average Navier-Stokes (RANS) models, with particular attention to the standard $\kappa - \epsilon$, $\kappa - \omega$-SST, and Reynolds stress models. Once we had our single-phase setup, we introduced a disperse phase by patching the flow field with a uniform initial concentration of oil. We updated the velocity field and, in the meanwhile, we also tested different drag force models, in

particular Schiller-Naumann (Shiller and Naumann, 1935) and Morsi-Alexander (Morsi and Alexander, 1972) to highlight if differences were present.

Solver

The solver type was set to Pressure-Based, velocity formulation absolute and transient simulations were performed. Gravity was added and set equal to -9.81 $\frac{m}{s^2}$ on the y-axis. Eulerian-Eulerian model was used to introduce the disperse phase, selecting 2 phases. The turbulence models used were: standard $\kappa - \epsilon$,- realizable, -rng, $\kappa - \omega$-SST and Reynolds stress with 7 equations; we did not use any near-wall treatment and the turbulence of the disperse phase was ignored.

Phases

Two phases were used to simulate water, as continuous phase, and silicone-oil, as disperse phase. Water was selected from Fluent database, while silicone-oil was added as an extra phase with the following properties:

Table 5.3: Physical properties of the different silicone oils tested in the ESCO Mixer. In particular, different viscosities and initial diameter were studied.

Name	Value
Density, kg/m^3	759 (low viscosity), 964 (high viscosity)
Viscosity, mPas	0.005, 12.3, 32.2, 242
Surface Tension, mN	0.0114
Initial Diameter, m	1.2e-5, 2.2e-5, 3.2e-5 5.5e-5

Cell Zones and Boundary Conditions

Two zones have been identified. The first zone includes the edges of the tank, the anchor and all the cells far from the impeller, while the second contains all the cells around the impeller and its radius is 120% the radius of the impeller. This second zone, named rotating region, has the origin of the axis in (0.0488, 0.043225, 0) and the rotation axis vector coordinates are (0.165, 0.986,0). Regarding the boundary conditions, we tested both fixed and moving shaft according to the rotating region reference, while all the walls are considered as fixed wall, and the top of the tank is a fixed wall with zero shear.

Numerical Schemes and Under Relaxation Factors

All the solution methods are then reported in Table 5.4:

Table 5.4: Computational methods and numerical schemes adopted in the simulation of ESCO Mixer for all the different viscosities.

Variable	Method
Pressure-veocity coupling	SIMPLE
Gradient	LSQ
Momentum	Second Order Upwing
Volume Fraction	QUICK
Turbulent Kinetic Energy	Second Order Upwind
Turbulent Dissipation Rate	Second Order Upwind
User Scalar 0	First Order Upwind
User Scalar 1	First Order Upwind
User Scalar 2	First Order Upwind
User Scalar 3	First Order Upwind
User Scalar 4	First Order Upwind
User Scalar 5	First Order Upwind

While under relaxation factors are reported in Table 5.5:

Table 5.5: Under relaxation factors for all the quantities of ESCO Mixer testcase.

Variable	Under-Relaxation Factor
Pressure	0.3
Density	0.3
Body Forces	0.5
Momentum	0.3
Turbulent Kinetic Energy	0.3
Turbulent Dissipation Rate	0.3
User Scalar 0	0.3
User Scalar 1	0.3
User Scalar 2	0.3
User Scalar 3	0.3
User Scalar 4	0.3
User Scalar 5	0.3

Initialization

In order to avoid divergences and problems with the solution of the population balance equation, it is important to calculate *a-priori* the initial value of all the moments, starting from the following definition:

$$\alpha_{phase2} = k_v M_3 = k_v M_0 \exp\left(3\mu_{log} + 4.5\sigma_{log}^2\right) \tag{5.3}$$

Where α_{phase2} is the volume fraction of the disperse phase, k_v is a shape factor and for spherical particles its value is $\pi/6$, M_0 and M_3 are the moments of order zero and three of the droplet size distribution, divided by the volume fraction of the disperse phase ($M_0 = \frac{m_0}{\alpha_{phase2}}$), while μ_{log} and σ_{log} are calculated together with the initial values of the moments, assuming a log-normal distribution of the equilibrium diameter (d_{32}):

$$M_n = \exp\left(n\mu_{log} + n\frac{1}{2}n^2\sigma_{log}^2\right), \tag{5.4}$$

$$\mu_{log} = \log\left(\frac{d_{32}^2}{\sqrt{\nu + d_{32}^2}}\right), \tag{5.5}$$

$$\sigma_{\log} = \sqrt{\log\left(\frac{\nu}{d_{32}}^2 + 1\right)}, \tag{5.6}$$

Where n is the order of the n-th momentum and it ranges from zero to five, d_{32} is the Sauter diameter, and $\sqrt{\nu}$ is the standard deviation that can be assumed as 15% of the initial Sauter diameter of the droplets. In our specific case, for the low viscosity oil, the initial volume fraction is equal to 0.0131 and the initial values of the moments are reported in Table 5.6:

Table 5.6: Initial values of the moments according to a log-normal distribution for the case of 1.31%wt of silicone-oil (low viscosity) in water and 1% for the other viscosities.

Moments	**0.5mPas**	**12mPas**	**34mPas**	**242mPas**
M_0	1.03×10^{15}	1.69×10^{14}	5.46×10^{13}	5.46×10^{13}
M_1	1.24×10^{10}	4.07×10^{9}	1.91×10^{9}	1.91×10^{09}
M_2	1.52×10^{5}	9.98×10^{4}	6.84×10^{4}	6.84×10^{4}
M_3	1.91×10^{0}	2.50×10^{0}	2.50×10^{0}	2.50×10^{0}
M_4	2.45×10^{-5}	6.42×10^{-5}	9.37×10^{-5}	9.37×10^{-5}
M_5	3.22×10^{-10}	1.68×10^{-9}	3.58×10^{-9}	3.58×10^{-9}

It is important that during the whole simulation the value of M_3 is kept constant. This value is indeed related to the volume fraction of the system and it will be proved in the reminder of this work. This means that, in order to ensure the conservation of the mass along the run, the value of M3 can be monitored and reported. If this value changes in time, this means that mass is lost, and boundary conditions or setup must

be checked. Velocity field is loaded as previously developed and PBE can be solved in a segregated way.

This means that the equations of moments are solved while the flow field is frozen. This is possible only because the volume fraction of the disperse phase is low and the size of the droplets does not influence the viscosity of the system. In cases where the volume fraction is higher or the viscosity depends on the DSD, it is necessary to solve moments together with the flow field, at every single timestep. The residuals were constantly monitored, and the convergence value was set equal to 1×10^{-8} for all the moments. For each simulation, more than four thousand physical seconds were simulated, and the value of each timestep was set equal to 1. For each timestep, at least 250 iterations were performed.

5.5.2 Silverson Mixer

Computational Domain

Silverson mixers belong to rotor-stator mixers. The specific 150/250 model is composed by an impeller with two sets of blades (resp. 4 and 8 blades) and two screens (stator), as it is possible to appreciate in figure 5.12:

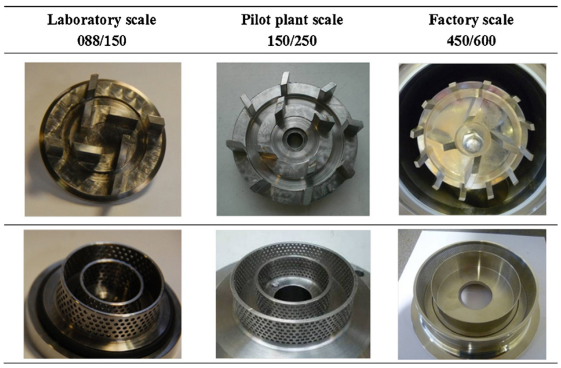


Figure 5.12. *Three models of Silverson mixer (from left to right: lab, pilot, plant scale) used in personal/home care manufacturing as an ending part of the mixing process to furhter reduce the size of the droplets. In particular, it is possible to observe how the number and dimension of the holes of the screens change together with the number of blades of the different impellers.*

The three-dimensional representation of the pilot version of a Silverson mixer can be appreciated in figure 5.13:

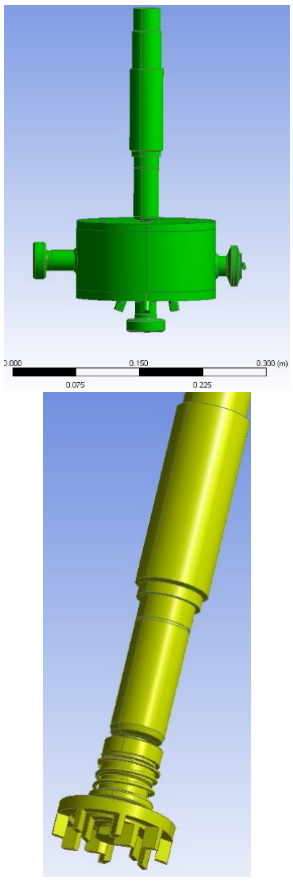


Figure 5.13. *3D CAD of the Silverson Mixer, pilot plant scale, obtained with Design Modeller. On the left: overview of the CAD model. On the right: detail of the impeller of the Silverson Mixer. The impeller is composed by two series of blades that move together.*

Some of the details, such as the springs that form the impeller shaft and the screws have been replaced by simpler geometries, because they are not affecting the fluid dynamics of the system but, on the contrary, are extremely difficult to mesh. The most important part of our CAD geometry is represented by the impeller (corona and shaft) and the two screens.

Computational Details

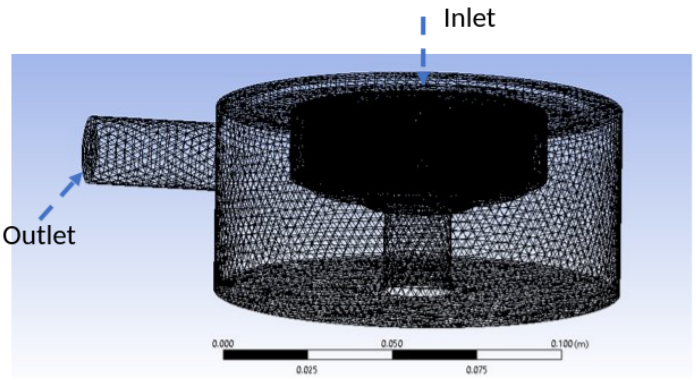


Figure 5.14. *Example of one Mesh of the Silverson Mixer, composed by almost 3 000 000 of tetrahedrons. An high number of elements is required to model the numerous holes present in the two screens and the impeller which is located in their proximity. The mixer is composed by an inlet and an outlet pipe.*

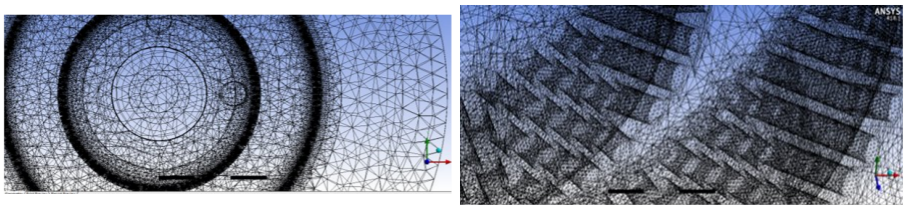


Figure 5.15. *Details of the Mesh. In particular, it is possible to observe the rotating zone surrounding the two screens and the high number of holes that require most of the elements composing the mesh.*

Solver

The solver type was set to Pressure-Based, velocity formulation absolute and transient simulations were performed. Gravity was added and set equal to -9.81 $\frac{m}{s^2}$ on the y-axis. Eulerian-Eulerian model was used to introduce the disperse phase, selecting 2 phases. The turbulence models used were: $\kappa - \epsilon$,- realizable, -rng, $\kappa - \omega$ SST; we did not use any near-wall treatment and the turbulence of the disperse phase was ignored.

Phases

Two phases were used to simulate water, as continuous phase, and silicone-oil, as disperse phase. Water was selected from Fluent database, while silicone-oil was added as an extra phase with the following properties:

Table 5.7: Physical properties of the different silicone oils tested in the Silverson in-line rotor-stator Mixer.

Name	Value
Density, kg/m^3	964
Viscosity, mPas	9.90
Surface Tension, mN	0.0114
Initial Diameter, m	5.5e-5

Cell Zones and Boundary conditions

Two zones have been identified. The first zone includes the edges of the tank and the cells surrounding from the screens, while the second contains the interior region from the external screen to the internal chamber. This second zone, named rotating region, has the origin of the axis in (0.0, 0.0, 0.0) and the rotation axis vector coordinates are (0.0, 0.0,1). Inlet boundary was set to mass-flow rate and the % of disperse phase was specified as inlet condition, while the outlet is a pressure-outlet. Walls have been modelled with no slip condition.

Numerical Schemes and Under-relaxation Factors

Numerical Schemes and URF are similar to those used for the simulation of ESCO 6l and can be found respectively in Table 5.4 and Table 5.5

Initialization

Inlet flow is composed by two fluids, water and 1% silicone-oil with viscosity equal to 9mPas and initial size of the droplets equal to 5e-5m. Initial values of the moments have been retrieved as in the ESCO 6L case.

Table 5.8: Initial values of the moments according to a log-normal distribution for the case of $1\% wt$ of silicone-oil in water.

Moments	Initial Value
M_0	1.17×10^{13}
M_1	6.24×10^{8}
M_2	3.42×10^{4}
M_3	1.92×10^{0}
M_4	1.09×10^{-4}
M_5	6.39×10^{-9}

Chapter 6

Results

6.1 Introduction

In this chapter, the results concerning Copolymers and the mesoscale simulations are presented in the first part. The second part contains all the findings regarding emulsions and the simulations of the equipment scale. In the final part, the connection between the two scales is discussed in details.

6.2 Copolymers

6.2.1 Equilibrium Simulations

Coarse grained simulations were performed to understand the behaviour of complex fluids in equilibrium conditions. Different tests were made in this work, to ensure correctness of the simulation setup and the adherence to the experiments.

Equilibrium and non-equilibrium simulations were performed in order to obtain information regarding the microstructures that are formed within a complex fluid, but also their evolution in time when a flow field is acting on the system. This information becomes extremely important, for such systems, also at the industrial scale because transport properties are affected by the microscopic behaviour of the system.

6.2.2 Phase Diagrams

The first step was to perform simulations on boxes containing only water, and then on complex systems composed by Pluronics L64, P85, P104 a surfactant, and water at different concentrations. Different boxes, timesteps, concentrations were tested in order to reduce the number of possible artefacts related to the confined geometry. The aim of this part of the work was to obtain reliable phase diagrams, where polymeric chains interact between themselves and the surrounding water molecules, to track all

the possible microphases and peculiar structures that are experimentally observable. We performed simulations over all the spectrum of concentrations of Pluronics L64, P85 and P104 in water (i.e. from 5% to 95%) within different simulation boxes and we extracted quantitative and qualitative information regarding the equilibrium phases. We identified a box size that is a good compromise between simulation time and reduced number of artefacts to perform the following batches of simulations. Experimental phase diagram for the water/Pluronics L64 system at equilibrium was found in the literature and compared against the phases obtained from the simulations, performed in LAMMPS. In Figure 6.1, snapshots of water/Pluronics L64 mixture at different concentrations are reported.

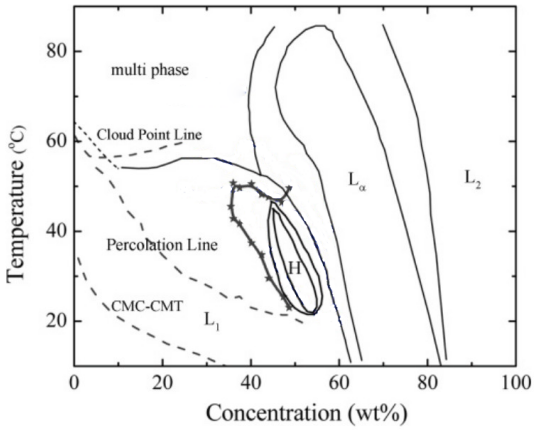


Figure 6.1. *Experimental phase diagram (Zhou et al., 1996) for the mixture of Pluronics L64/Water mixture showing the different phases that are obtained by varying the temperature and the concentration of the Pluronics L64.*

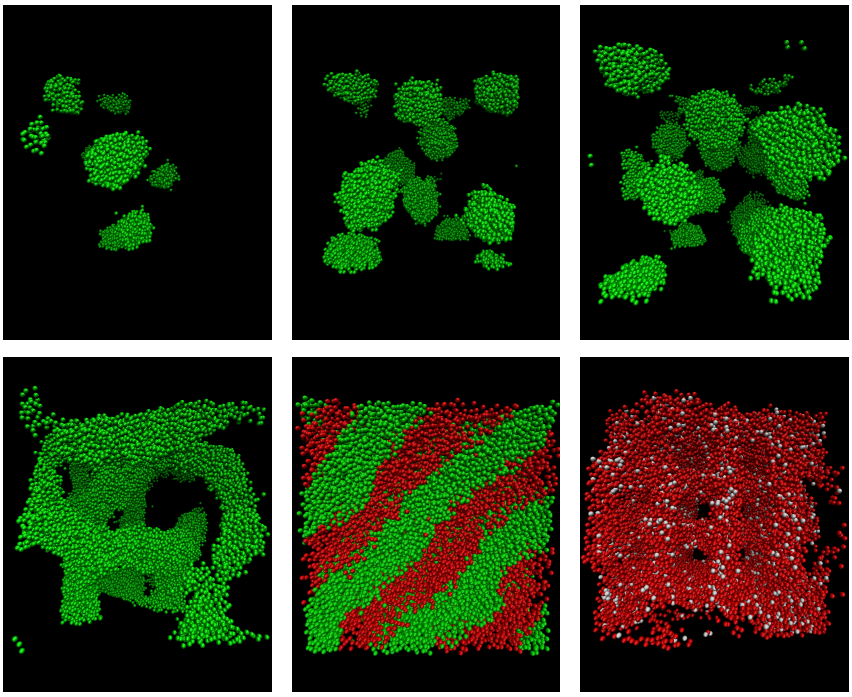


Figure 6.2. *From left to right, top to bottom: Graphical outputs of the simulation results of Pluronic L64 in Water at increasing concentrations (5%, 15%, 25%, 45%, 75%, and 90% wt) where all the experimental phases can be recognized. In particular, micelles, worm-like, lamellae, and reversed micelles can be observed at different concentrations. In each snapshot, different colours represent different beads (PPO beads: green, PEO beads: red) and in many cases water was faded for the sake of clarity (Droghetti et al., 2018).*

The reference temperature for this system was assumed to be $298K$, which corresponds to $1k_BT$ in DPD units. For the sake of brevity, only the results obtained by using the $30 \times r_c$ box are reported in this case, but similar results were obtained with bigger box. The simulation time, composed by around 1 million timesteps, lies between 4 and 9 hours ($30 \times r_c$ box), 20 and 30 hours ($40 \times r_c$ box), based on the number of interactions that have to be computed. The density of the DPD system was set equal to 3, meaning that there are 3 beads per unit volume (e.g. for a box size equals to $30 \times 30 \times 30$, there are 81000 beads). Different concentrations have been obtained by varying the number of water and Pluronics L64 particles. Pluronics L64 beads are described as molecules, hence the number of chains representative of the desired concentration, was used instead of the number of beads. A similar approach was tested with other types of Pluronics, i.e. Pluronics P104 (Álvarez-Ramírez et al., 2009) and P85, in order to validate the previous setup. In Figure 6.4, we reported the different microphases that have been identified for this system, while in Figure 6.3, these phases are matched with the experimental phase-diagram. In particular, several phases were found in our simulations that matches with the experimental evidences. In Figure 6.4

it is possible to appreciate micelles (a), elongated micelles (b), hexagons (d), lamellae(f) and mixed phases(g). These results were obtained by varying the concentration of Pluronics P104 into the same simulation box and running equilibrium simulations. Similar results were obtained for the Pluronics P85 (Figures 6.5 and 6.6), where the simulation parameters were kept constant but the length of the polymeric chain. Again the phase diagram has been explored.

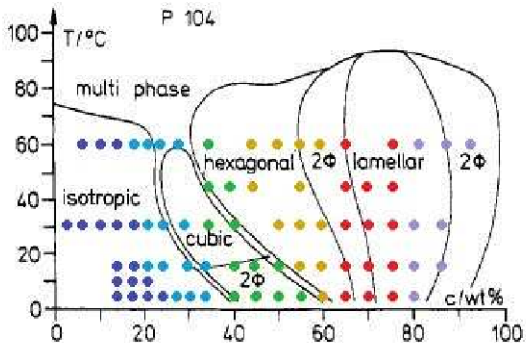


Figure 6.3. *Experimental phase diagram (Álvarez-Ramírez et al., 2009) for the mixture of Pluronics P104/Water mixture showing the different phases that are obtained by varying the temperature and the concentration of the Pluronics P104. Dots represent the simulations that have been performed. Colors represent the different phases that have been identified in our simulations: blue - isotropic, light blue - elongated micelles, green - cylinders, yellow - hexagonal, red - lamellar, grey - two phases.*

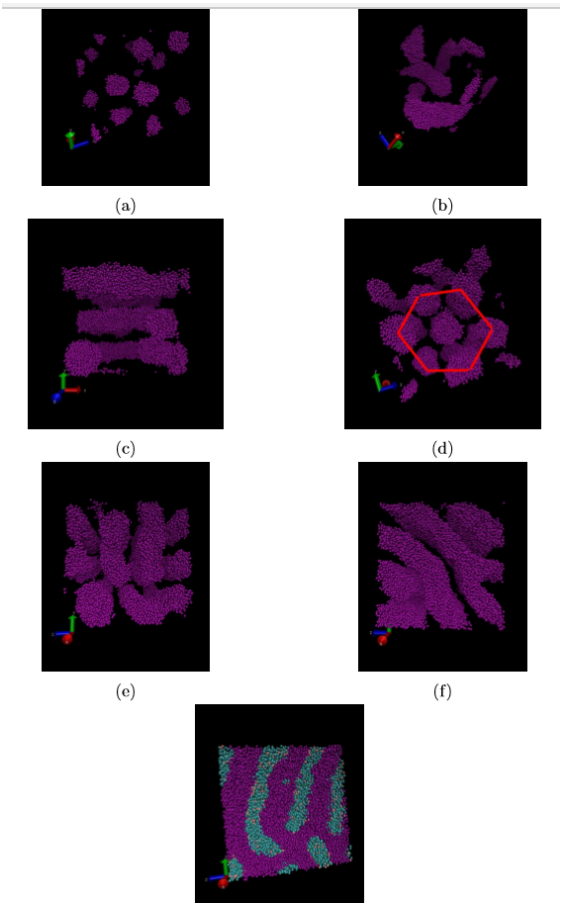


Figure 6.4. *From left to right, top to bottom: Graphical outputs of the simulation results of Pluronics P104 in Water at increasing concentrations (*5%, 15%, 25%, 40%, 60%, 75%, *and* 90% *wt) where all the experimental phases can be recognized. In particular, micelles, worm-like, hexagons, lamellae, and reversed micelles can be observed at different concentrations. In each snapshot, different colours represent different beads (PPO beads: purple, PEO beads: cyan, Water: pink) and in many cases water was faded for the sake of clarity.*

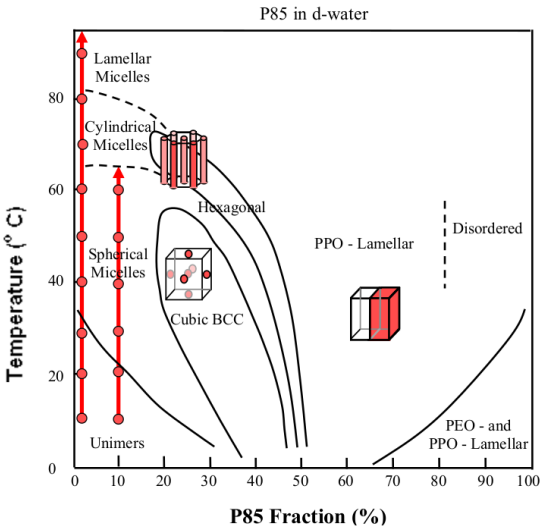


Figure 6.5. *Experimental phase diagram (Hammouda, 2010) for the mixture of Pluronic P85/Water mixture showing the different phases that are obtained by varying the temperature and the concentration of the Pluronic P85. Dots represent the simulations that have been performed.*

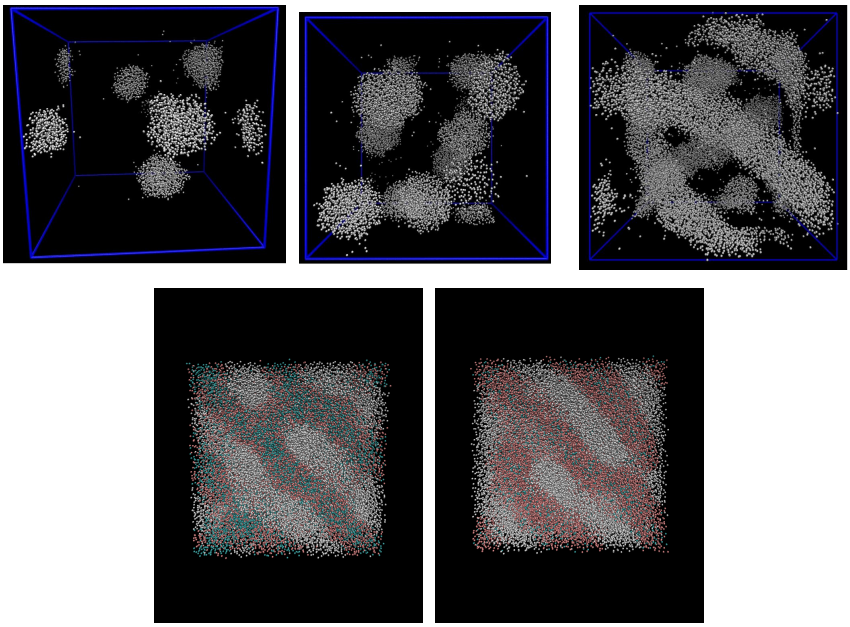


Figure 6.6. *From left to right, top to bottom: Graphical outputs of the simulation results of Pluronic P85 in Water at increasing concentrations (10%, 25%, 50% 70% 90% wt) where all the experimental phases can be recognized. In particular, micelles, worm-like, and lamellae, can be observed at different concentrations. In each snapshot, different colours represent different beads (PPO beads: grey, PEO beads: pink, Water: cyan) and in many cases water was faded for the sake of clarity.*

They can be compared with their experimental diagrams (Álvarez-Ramírez et al., 2009; Hammouda, 2010; Zhou et al., 1996). The set of parameters used to simulate P104 and P85 was equal to the parameters used for L64, while the equilibrium phase diagram was obtained via SANS. Difficulties in finding the cubic phase were related to the confined phase and the impossibility of the model of describing such interacting systems. Only repulsive forces may not be enough to represent ordered structure the cubic system. Also, bi-phases were difficult to obtain due to the uncertainty of labelling their structure from a qualitative point of view. Different temperatures were obtained, in this specific case, by varying the k_BT parameter in each LAMMPS simulation. For the P104, the obtained phase diagram, corresponding to $k_BT = 1$ (around 330 K), shows a clear match between simulated and experimental phase boundaries. A small deviation of the lamellar phase to lower concentrations can be also appreciated (i.e. lamellae are found at lower concentrations than expected).

After having assessed which phases we were able to simulate, we wanted to quantify, as an additional element for comparison, the sphericity of the micelles, that were obtained in the simulation and compare those results with the predicted phase in the experimental phase diagram. As it was described in the previous section, we tested different boxes first. A qualitative overview is reported in Figures 6.7, 6.8 and

6.9, where similar concentrations are reported for different box sizes. A total of 300 simulations were performed at this stage, to validate the entire phase diagram.

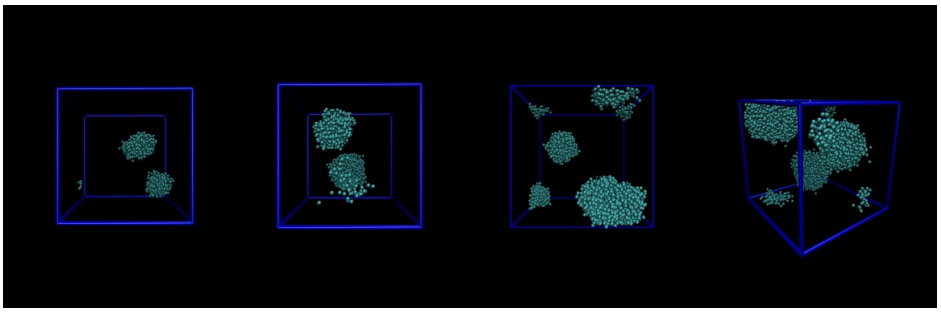


Figure 6.7. *From left to right: Pluronic L64/water mixtures at different concentrations (5%, 10% 15%, and 25% wt) in a simulation box of* $20 \times r_c$*. Spherical micellar structures can be identified.*

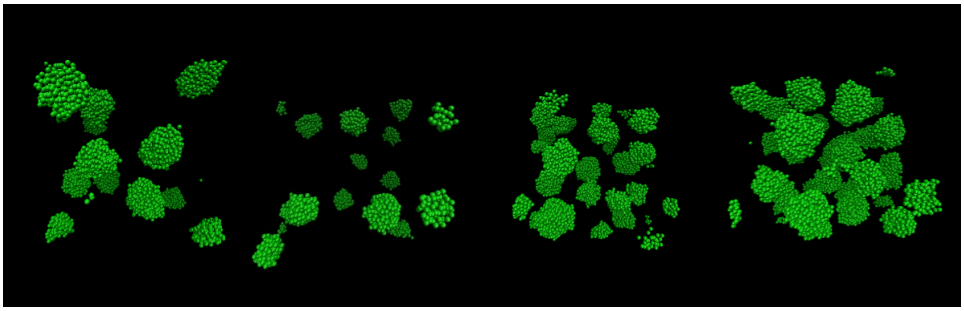


Figure 6.8. *From left to right: Pluronic L64/water mixtures at different concentrations (5%, 10% 15%, and 25% wt) in a simulation box of* $30 \times r_c$*. Spherical micellar structures can be identified.*

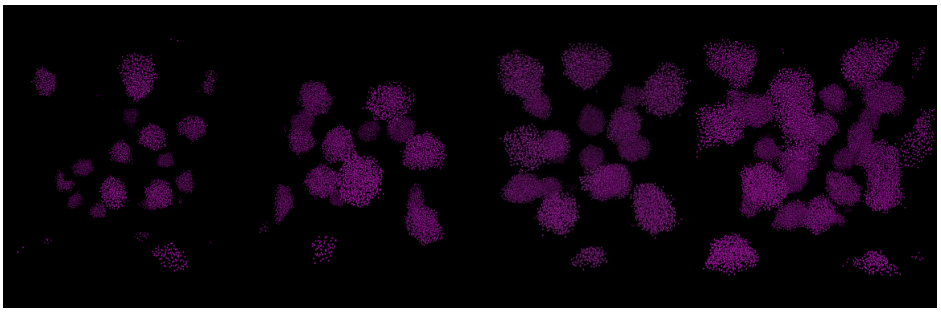


Figure 6.9. *From left to right: Pluronic L64/water mixtures at different concentrations (5%, 10% 15%, and 25% wt) in a simulation box of* $40 \times r_c$*. Spherical micellar structures can be identified.*

6.2.3 Cluster Analysis

We chose a set of simulation parameters (e.g. box size, timestep, end time…) in which the number of micelles formed by Pluronics L64, P85 and P104 was enough to have statistical results but also reduce computational time either for the clustering algorithm and the simulation code. Regarding the clustering algorithm, already introduced in the previous Chapter, we collected coordinates of each PPO bead for every timestep and checked whether or not a bead was part of a cluster. The information we obtained from such algorithm was both the number of clusters at every timestep, then averaged through all the simulation time, and the size of each cluster. Having these number, it is possible to calculate a scaling exponent that is related to the shape of the clusters. This analysis was performed at low concentrations, where the number of clusters can be clearly defined, while for higher concentrations, the algorithm fails at finding separated structures, due to the interconnections between the polymeric chains. Also, periodic boundary conditions were used for low concentrations, when the number of particles was reasonable, to limit the effects of boundaries.

This was done because when a cluster crosses a boundary, if PBC is not properly set up, it is counted two times as two halves of the initial cluster. Also, the boundary crossing of few beads during one single timestep, can create noise for similar reason.

We knew that nine beads of PPO form a chain, so we recorded the number of chains contained in each cluster and stored that value. The cluster analysis was performed on the $30 \times r_c$, without filtering any timestep, by reporting the calculated gyration radius against the aggregation number, meaning that each dot in Figure 6.10, represents a single cluster containing N chains and with a gyration radius equals to R_g. The calculation of the gyration radius can be used as indicator to understand the shape of the clusters and this aspect will be discussed in the reminder of this work. However, it is already possible to appreciate how the size of the clusters increases with the concentration, hence a shift in the phase diagram from spherical to elongated micelles. It must be highlighted that coordinates from LAMMPS are not saved every timestep. We collected coordinates every 200 timesteps, otherwise both the dimension of the log file and the clustering time would have been too difficult to handle. Also, a tuning on the skin parameters, illustrated in the description of the Python algorithm, was performed for all cases in order to ensure that all the beads that were present at the beginning of the simulation are found (i.e. conservation of mass is respected). Skin parameter ensures that all the interactions are taken into account. This means that if this value is too small some of the particle-particle interactions are lost. The tuning of this parameter was done in an empirical way, meaning that it was ensured mass conservation by reducing the skin value. It is possible to observe that at low conentration, most of the aggregates have spherical shape (trend is similar to the red line) while they become elongated structures at higher concentrations (e.g. 25%) where bigger aggregates appear in the simulation box.

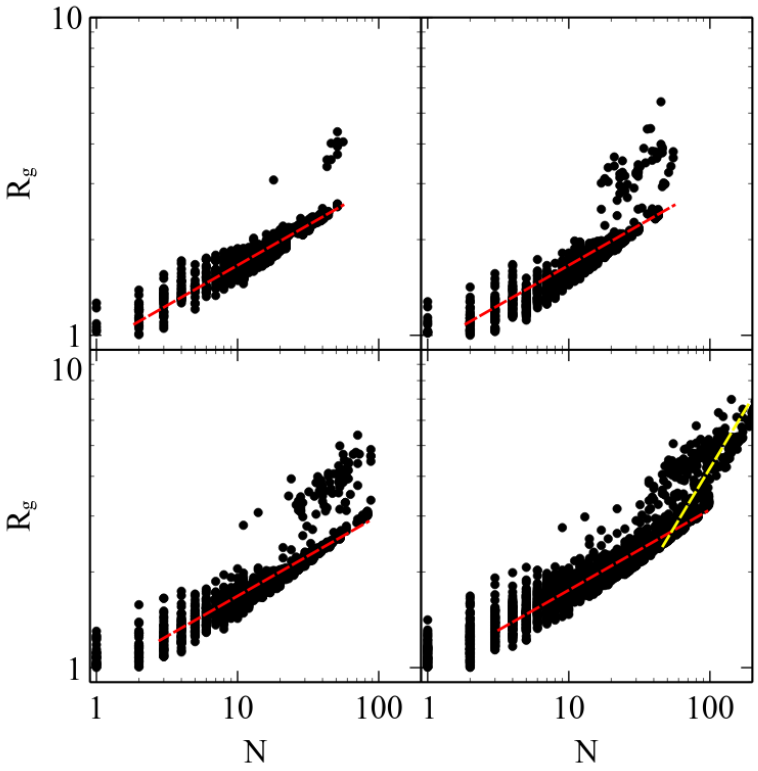


Figure 6.10. *Gyration radius, R_g, is reported against the aggregation number, N, for four concentration of Pluronic L64 in water (from left to right, top to bottom: 5%, 10%, 15% and 25%). The red line indicates a slope of 0.3 while the yellow line a slope of 0.5. Clusters have been identified by using the cluster algorithm developed in this work.*

In this part, the results of the cluster analysis are reported. In Figure 6.11 and 6.12 it is possible to appreciate how the number of clusters is reported against their aggregation number. This approach can be used to estimate the shape of the aggregates that are present into the simulation box. We performed cluster analysis by varying the concentration and type of Pluronics into the simulation box. The result of this analysis was the calculation of the ration between the gyration radius and the aggregation number. When this parameter is equal to 0.3, aggregates are spherical. We proved that at low concentrations for both the Pluronics, aggregates are spherical micelles. However, when the concentration of the Pluronics becomes higher and higher, this number varies and the aggregates undergo a phase transition. Table 6.1 and Table 6.2 report for each concentration the ratio between the gyration radius and the aggregation number for Pluronics P64 and P104.

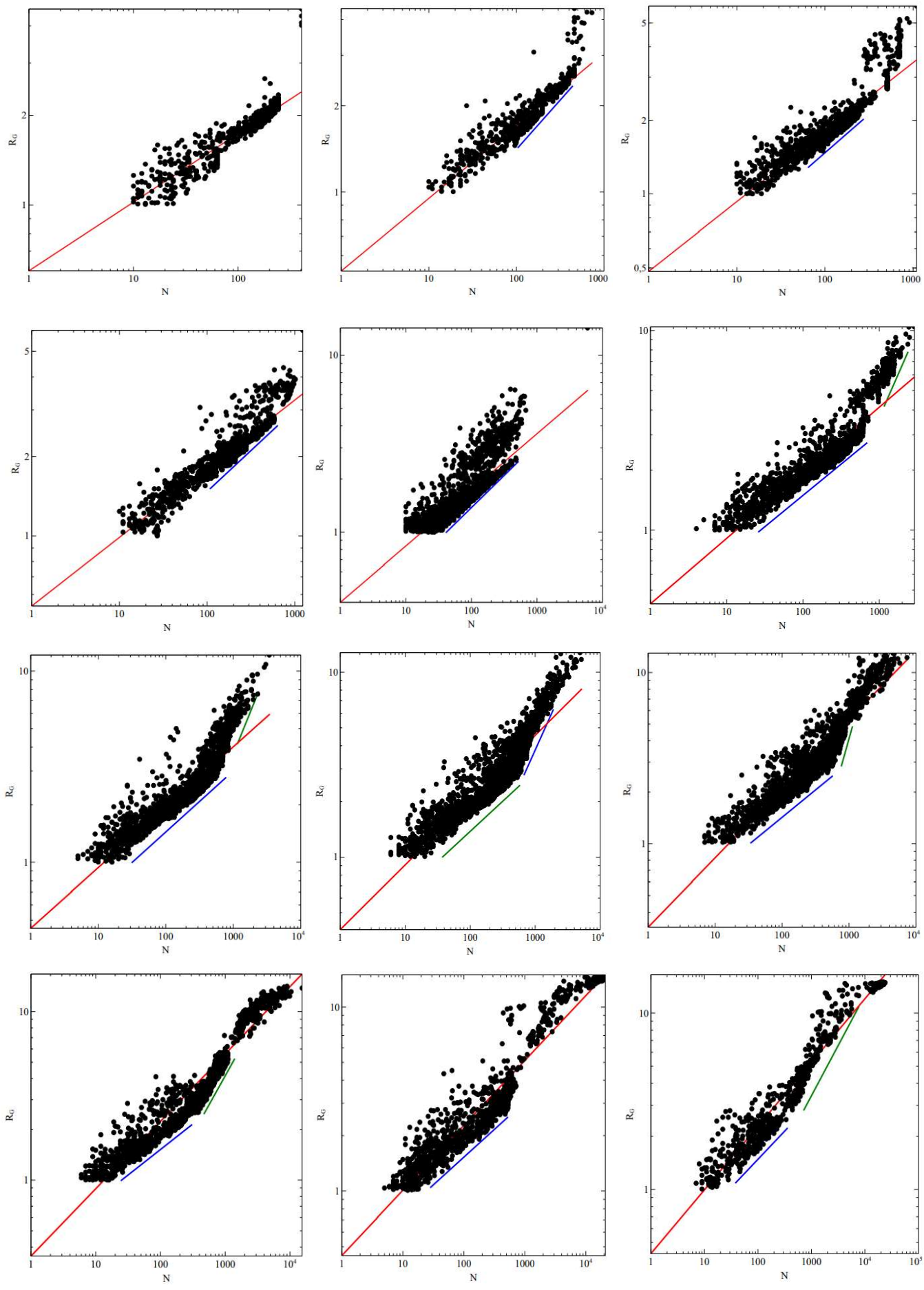

R_G
N

Figure 6.11. *L64 Cluster analysis - From left to right, top to bottom: Gyration radius is plotted against the number of chains contained into a single cluster. The concentrations are: 3%, 5%, 8%, 10%, 15%, 20%, 25%, 30%, 35%, 40%, 45%, 50% of Pluronic in water. Red line is the slope of the trend of the structures, while the blue line indicates the limit of the cloud of points*

Table 6.1: d coefficient of Eq. 5.2 calculated for the main lines (red lines in Fig 6.10 and similar) and boundary lines (blue-green) for L64

Concentration	R_G/N
0.03	0.231
0.05	0.254
0.08	0.284
0.10	0.260
0.15	0.317
0.20	0.329
0.25	0.316
0.30	0.350
0.35	0.403
0.40	0.400
0.45	0.353
0.50	0.361

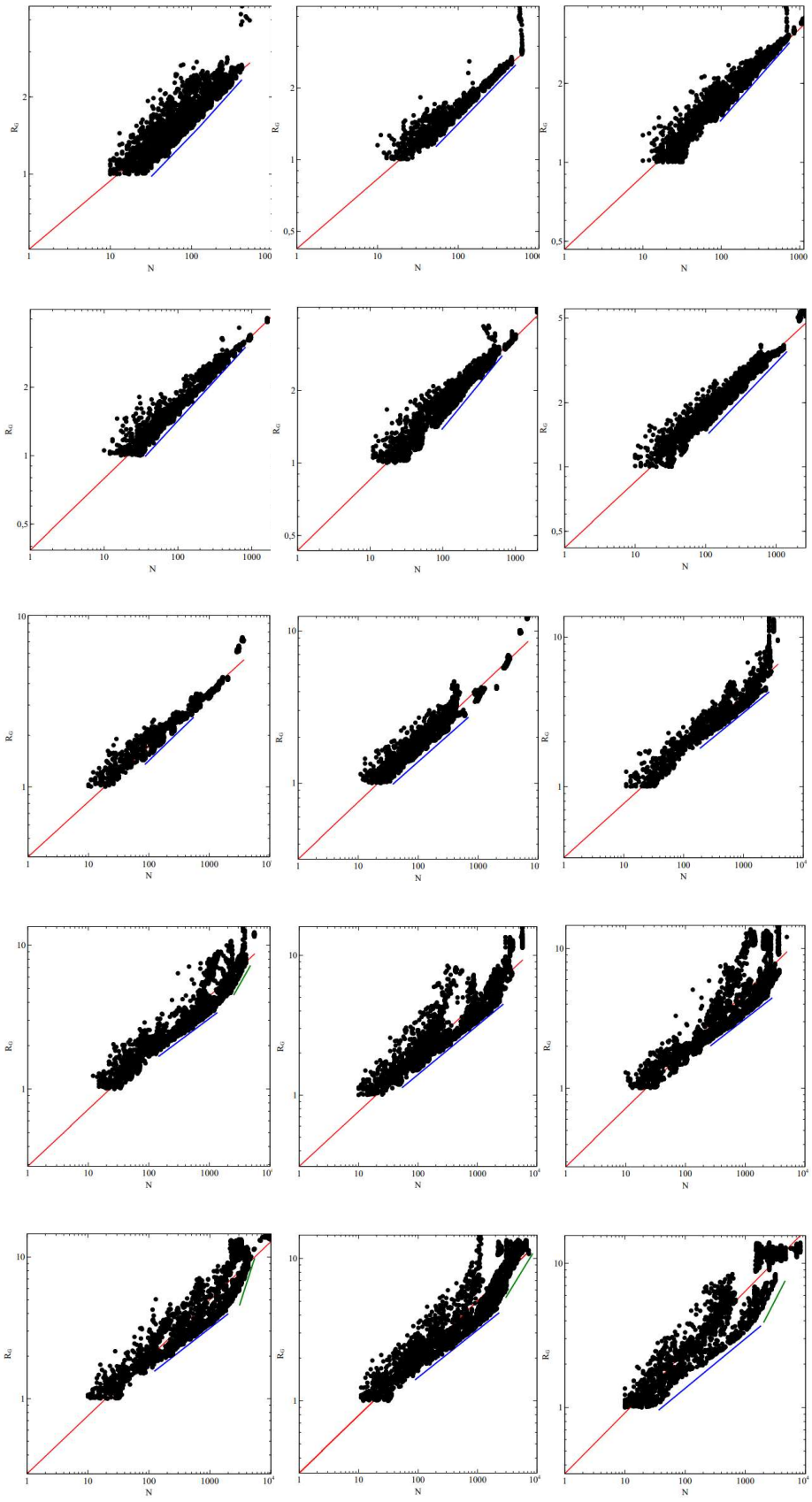

R_G
N

Figure 6.12. *P104 Cluster analysis - From left to right, top to bottom: Gyration radius is plotted against the number of chains contained into a single cluster. The concentrations are: 3%, 5%, 6%, 8%, 9%, 10%, 12%, 15%, 18%, 20%, 25%, 30%, 35%, 40%, 45%, 50% of Pluronic in water. Red line is the slope of the trend of the structures, while the blue line indicates the limit of the cloud of points*

Table 6.2: *d* coefficient of Eq. 5.2 calculated for the main lines (red lines in Fig. 6.10 and similar) and boundary lines (blue-green) for P104

Concentration	R_G/N
0.03	0.267
0.05	0.293
0.06	0.287
0.08	0.281
0.09	0.314
0.10	0.295
0.12	0.308
0.15	0.322
0.18	0.372
0.20	0.361
0.25	0.394
0.30	0.392
0.35	0.415
0.40	0.410
0.45	0.402
0.50	0.422

6.2.4 Chemical Potentials

The relationship between chemical potential and cluster mass distribution that is reported in section 6.2.7 (Figures 6.29 - 6.31) reads as follows

$$\frac{\mu_1 - \mu_N}{kT} = \log\left(\frac{1}{X_1}\left(\frac{X_N}{N}\right)^{\frac{1}{N}}\right) \tag{6.1}$$

where X_N/N is the concentration of the aggregates or molecular structures containing N copolymer molecules obtained from the clustering algorithm. Eq. 6.1 allows therefore to calculate the chemical potential of a cluster (μ_N) from the cluster mass distribution (X_N). For cylindrical structures the following equation for the standard chemical potential can be assumed:

$$\mu_N^0 = \mu_\infty^0 + \frac{\alpha kT}{N^p} \tag{6.2}$$

where α is a positive constant which depends on the molecular interactions and p is a parameter which depends on the shape of the aggregates and it varies from 1 (linear clusters) to 0.5 (cylindrical clusters) to 0.33 (spherical clusters). Tests on Pluronic P104 and L64 have been performerd and results are reported in Figure 6.13:

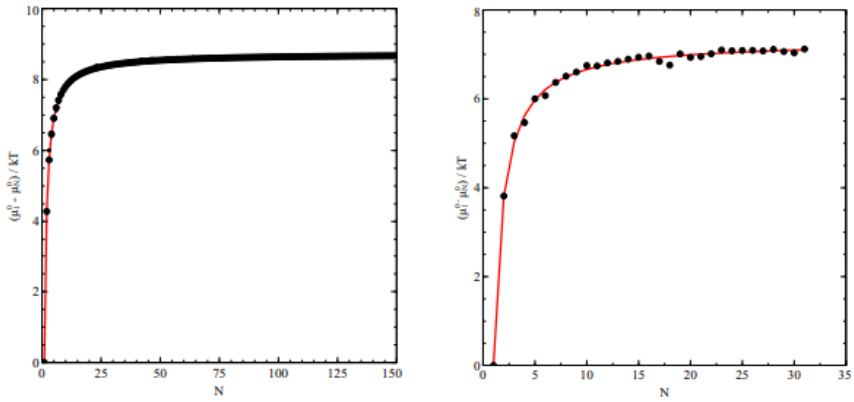
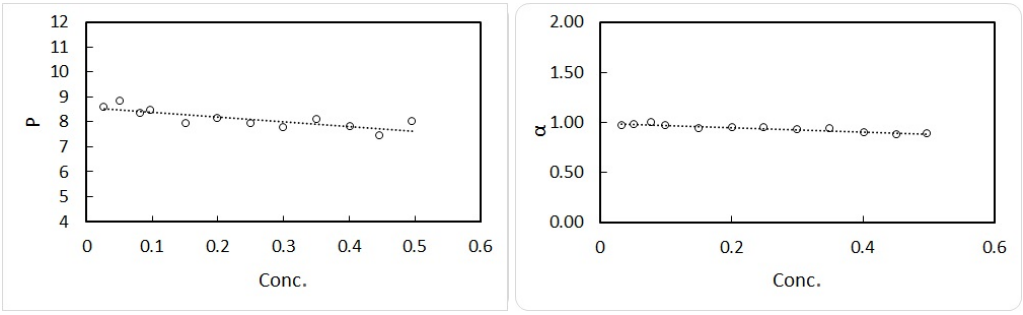


Figure 6.13. *Theoretical value of the chemical potential (red line) is plotted against the simulation results (dots) for Pluronics L64 (left) and Pluronics P104 (right).*

and the value of the parameter α and p can be obtained by least square optimization:



(a) *Value of the α parameter is reported against the concentration of Pluronics L64*

(b) *Value of the p parameter is reported against the concentration of Pluronics L64*

(a) *Value of the α parameter is reported against the concentration of Pluronics P104* **(b)** *Value of the α parameter is reported against the concentration of Pluronics P104*

The two parameters p and α are an indicator of the shape of the aggregates. The p parameter decreases while the concentration increases, because the shape of the aggregates moves towards more complex and elongated structures. The α parameter is constant and related to the concept of CMC of the copolymers in water.

6.2.5 Non-Equilibrium Simulations

It is important to understand what happens when equilibrium configurations undergo strong flow field effects (Boek et al., 1997; Gentile et al., 2014). This is a very common example in mixing processes. Microstructures can be deformed, re-oriented or simply transported by high shear regions. When an equilibrium structure is affected by shear stresses, its fate is unknown and difficult to predict even at experimental scale. It becomes fundamental to have tools that are able to predict how they evolve in time and if a new quasi-stable configuration can be found. The way in which this analysis is done is via Non-Equilibrium simulations. In particular it is important to reproduce the physics of the system and to stick to a realistic representation, trying to keep all the relevant parameters of interest in bounded ranges. In MD, it is quite common to apply huge gradients of a desired property to investigate the response of the system. Similar approach is used in CG simulations, even though scalability of CG variables into real, physical quantities is still unclear. We run different testcases to understand the set of parameters that was able to reproduce the physics of the system composed by a mixture of water and Pluronic L64 under shear stresses.

6.2.6 Range of applicability

The first step was to identify the range of applicability of the shear in DPD units. In DPD, everything is represented in DPD units, meaning that a conversion standard for the units is missing. We started investigating different thermostats (Soddemann et al., 2003) and rheological response (Fedosov, Caswell, et al., 2008; Moshfegh and Jabbarzadeh, 2015; Townsend et al., 2017) for a system composed only by water-like DPD beads. As it is reported in Figure 6.16, temperature was found to be bounded for

different thermostats, using a very small timestep, filtering the streaming effect on the particle, that add extra energy to the system.

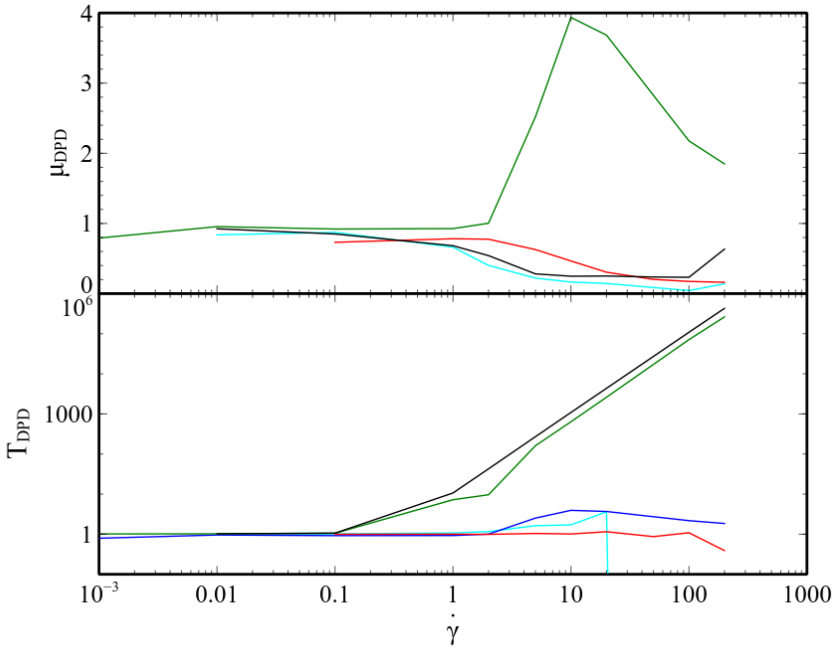


Figure 6.16. *Values of the viscosity (top graph) and temperature (bottom graph) in DPD units, obtained with LEBC in a box of* $30 \times r_c$*, a simulation time of 1 000 000 timesteps after 500 000 equilibration steps, with a 0.01* τ_{DPD} *using different thermostats (black: Berendsen, red: SLLOD, green: Langevine, blue: SLLOD + velocity ramp, cyan: Nose-Hoover)*

Anomalous patterns were identified for a system composed only by water-like beads, in the estimation of the rheological behaviour. In particular, we focused our attention only on the DPD thermostat, and evaluated the evolution of the viscosity in DPD units. We obtained an increased value of the viscosity when the shear range was greater than the unit, coupled with an increased temperature. This non-Newtonian behavior for the water, suggested us to limit the range of investigation to shear rates around the unity also for more complex systems.

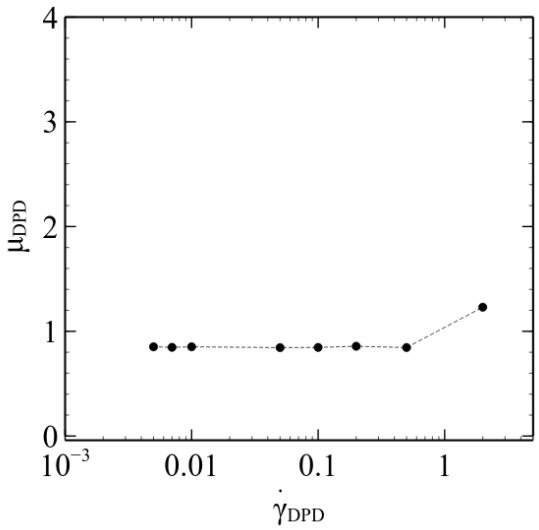


Figure 6.17. *Viscosity of a system containing water beads at different shear rates. The viscosity is constant in all the shear range explored, which indicates that the fluid behaves as a Newtonian fluid.*

In order to calculate the average viscosity, it is important that momentum is conserved across the simulation box. To validate this assumption, we computed the x-velocity profile along the y axis for different velocities, boxes and systems, as reported in Figure 6.18. Linear velocity profile, corresponding to a constant shear rate, can be reproduced in all our simulations, also thanks to the intrinsic nature of DPD that conserves momentum because all the hydrodynamic interactions are explicitly solved. This is a key element in our analysis that has been tested for all the following cases. From our analysis was also clear that a lower limit of investigation could have been identified. Low values of the velocities applied on the box, are not able to reproduce linear shear across the box, because they are masked by the temperature imposed on the particles, that is responsible for the thermal velocities of the beads. When thermal velocity competes with the shear velocity, the profile is not linear, meaning that momentum is not correctly transported across the box and the viscosity, and different bins have different shear rate. Also, the dissipative coefficient,γ , was equal for all the beads for all the cases. This is an important approximation because this coefficient can be seen as an artificial drag acting between the beads. We tested systems composed by two different species and different values of γ, and as expected, we found that velocity profile exhibits bending effects, meaning that a discontinuity is present.

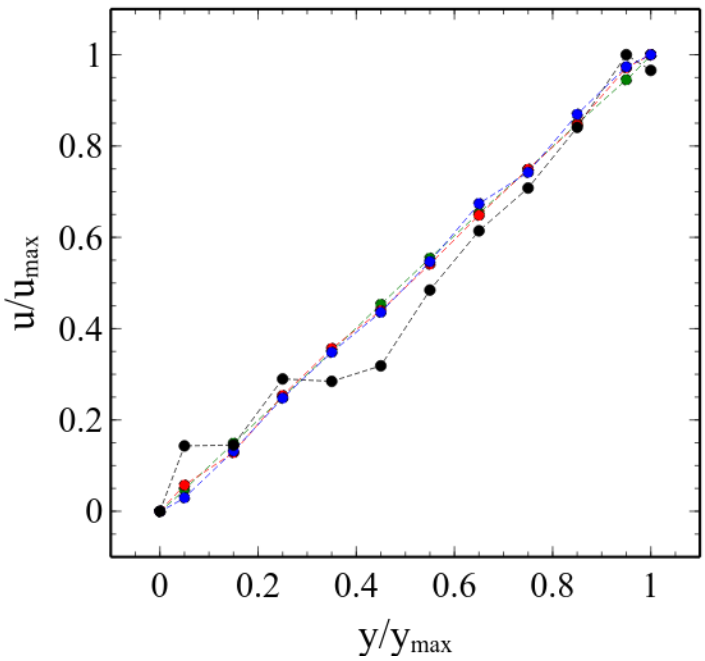


Figure 6.18. *Velocity profiles obtained acros the simulation box when LEBC are applied. The simulation box has a side length of* $30r_c$ *and it contains mainly water beads. Colors refer to different velocities obtained by varying the shear rate imposed on the system (black: 0.005, blue = 0.02, red = 0.2, green = 2.0). The axis have been normalized to the maximum dimension of the box and the velocity.*

After performing single type simulations, we moved to more complex systems under shear. We tested initially if the same conditions that we found acceptable for the single type, were still valid in the multicomponent system. As it is possible to see in Figure 6.19, we tested the velocity profile (e.g. $30 \times r_c$) at different timesteps, to have a clear understanding about how this value could have had an effect on the overall simulation. We also verified that the particles were uniformly distributed across the whole domain, by computing the density of particles across the vertical axis. This is a very important test, because it allows the detection of problems related to wrong boundary conditions. After these checks, we ensured that all the tested boxes, at different concentration of Pluronic L64 in water, have the same DPD density (beads per unit volume) and the velocity profile is completely developed along y-axis, even by varying the timestep. The timestep that we took from this analysis was 0.01 DPD units, below the limit proposed by Groot and Warren of 0.04, that is able to ensure good performances of the modified velocity Verlet Algorithm. This choice was related to the high shear stresses imposed on different systems, together with the neighbouring list that was updated and reconstructed after every single timestep. This was increasing the computational time for each simulation, but it was necessary because in cases of extreme shear, two close beads could move away from each other

of a distance greater than the cut-off length, in one time unit. With this test, we also ensured that the introduction, in the simulation box, of different chained beads does not interfere with the propagation of the momentum, since the dissipative coefficient was kept constant for all the interactions. One consideration, at this stage, could be represented by the fact that more equilibration steps are required in order to obtain a fully developed linear profile in mixed systems. Averages have been performed dividing the simulation box in bins. For every bin, the velocities of the beads have been averaged over 100 timesteps. A fully developed velocity profile was obtained after around 100 000 timestep for almost all the cases. Small deviations in the linearity of the velocity profile can be attributed to low values of the shear rate. This can be explained because the streaming effect is masked by the thermal motion of the beads (i.e. the temperature of the system is the expression of this motion). A remark must be done on the evaluation of the temperature of such systems. The application of the LEBC in LAMMPS is not directly implemented, but it has to be reproduced by using a combination of fixes. If the thermal velocity is computed as it is, results highlight an increased value of this parameter. This is due to the artificial streaming effect that can be seen as an introduction of energy into the system. Since this streaming effect is artificial and not natural, it has to be filtered. This means that the temperature should not be computed by using the real velocity of the beads, but the streaming velocity must be subtracted.

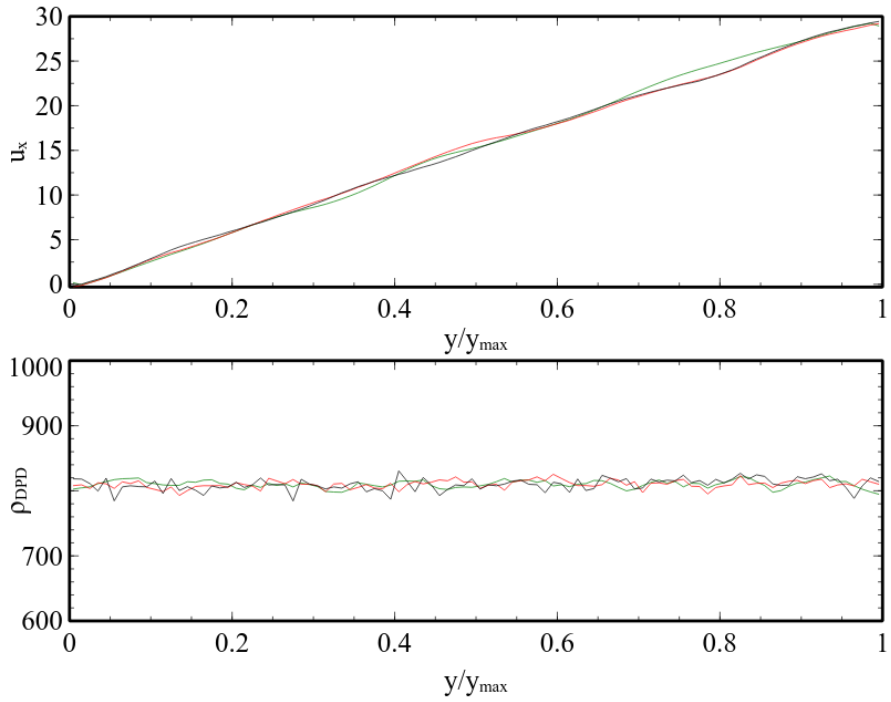


Figure 6.19. *On the top: velocity profiles developed across the simulation box in a system composed by Water and Pluronics L64. The shear rate imposed on the system is equal to 0.01 DPD units. On the bottom: density profile, i.e. number of beads per unit volume, along the y-axis. Similarly we found this profile on x-axis and z-axis. The uniform distribution of the beads in the box ensures no border effects. Colors refer to different timesteps (black = 0.001, red = 0.005, green = 0.01).*

6.2.7 DPD Viscosity

When performing coarse-grained simulation, it is important to avoid effects related to the limited dimension of the box. In fact, artefacts can be produced when dimensions are not big enough. Beads interact in confined spaces and boundaries effect can create anomalous structures that may be confused for real meso-phases. We tested three boxes in Non-Equilibrium simulations, and compared the obtained viscosity at different shear rates, as it can be observed in Figure 6.20.

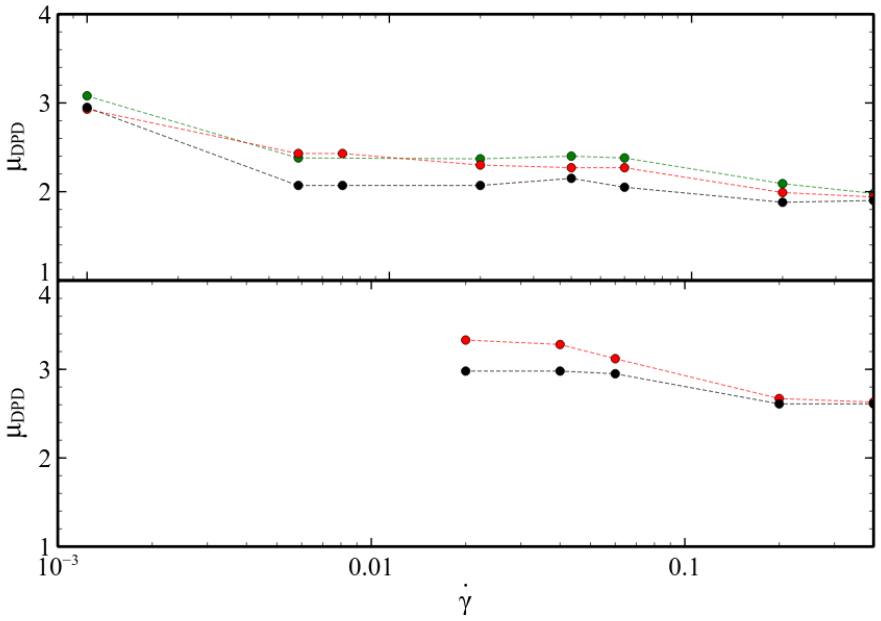


Figure 6.20. *Viscosity of 25%wt (top) and 35% (bottom) wt mixture of Plurnics L64 in water for different box sizes (black: 20× r_c, red: 30× r_c, green: 40× r_c) against the DPD shear rate.* 30× *and* 40× *curves are overlapping while small differences are obtained with the 20× box due to confinement effects.*

We observed that small differences can be appreciated between 20× and 30× r_c length, especially at higher concentrations of Pluronics L64, because polymer particles are forced to stay closer and even the phase-diagram results in not clear phases. However, for low concentrations, the difference between these three boxes can be neglected. In particular, no differences were found between 30× and 40× boxes. The first was selected as preferred testcase, because it provides a good compromise between validity of the results and reduced simulation time. We extended our simulations to all the range of concentrations of Pluronics L64 in water and computed the different viscosities, using the harmonic potential to connect beads in one polymer. We ensured for all different cases (shear and concentrations) that the velocities obtained in the simulation box were linear for all the systems, viscosities were extracted after plateau were reached at all the different shear rates. Figure 6.21 is an example of validation test for one specific concentration at different shear rates. It is possible to observe that after an initial phase in which the system orient itself because of the streaming effect, a constant viscosity can be recorded after 3×10^5 timesteps. Each final value for the viscosity was obtained by averaging every hundred timesteps, the value of the viscosity in the last thousands timesteps. Viscosity curves obtained in this way are in most cases flat and warning messages in LAMMPS started to highlight that extreme elongation was taking place between beads in polymeric chains.

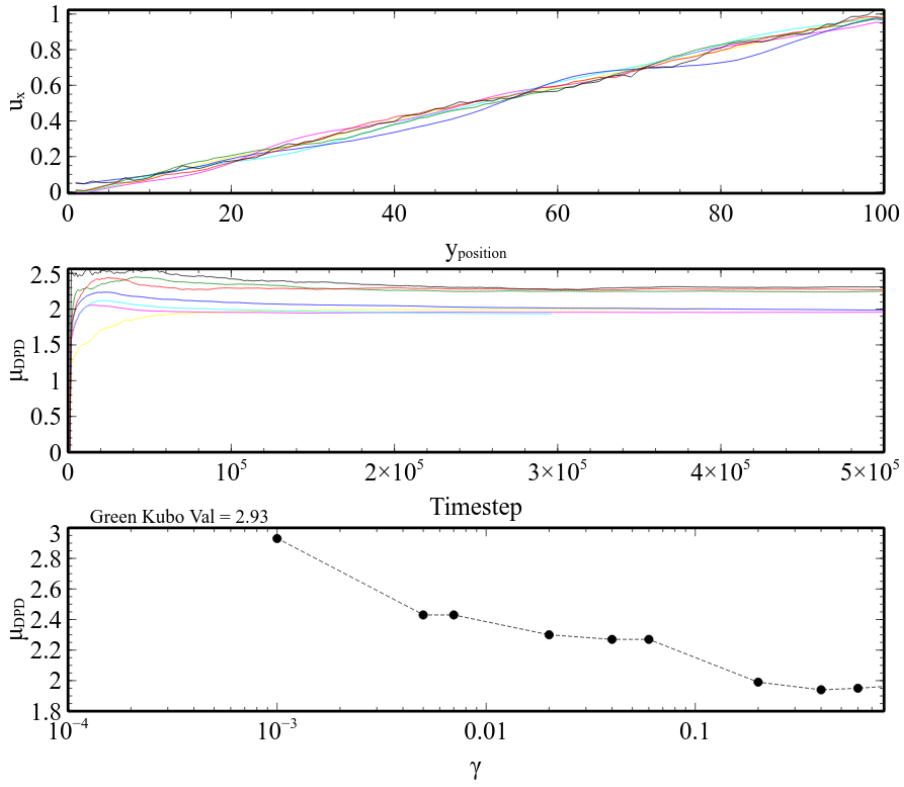


Figure 6.21. *Velocity profile (top) at different shear rates (black: 0.005, blue: 0.007, red: 0.01, green: 0.02, cyan: 0.05, magenta: 0.1, yellow: 1.0); value of the viscosity (middle) recorded over the simulation time until a plateau has been reached in all the cases for different shear rates (similar to top). Final value of the viscosity (bottom) reported against the DPD shear rate. All the cases refer to a box containing 25% wt of Pluronics L64 in water in a box with side length equals to 30× r_c*

So we focused our attention on solving the problem of over-elongated chains. We started our simulation by varying the harmonic constant in order to make the chains stiffer. We tuned the value of the κ_{ijharm} in order to reduce the variation in the calculated viscosity. When the chains undergo extreme shear, a weak constant could cause problems because beads overextend. In this way, we ensured that there is a minimum value of the harmonic constant such that further increasing does not change the profile of the viscosity curve. We tuned and tested this parameter on different concentrations, and in Figure 6.22 is reported the variation of the viscosity as a function of the shear rate for different harmonic constant. The cut-off radius for these forces was left equal to 1.0, even if further tests were performed also on this value. No differences were found when the cut-off value was in the range of 0.5 – 1.5 DPD length units. We kept the value of the cut-off equals to 1.0 DPD units. We focused our attention only on one box size, and the a timestep of 0.01 DPD units was used for

all the simulations.

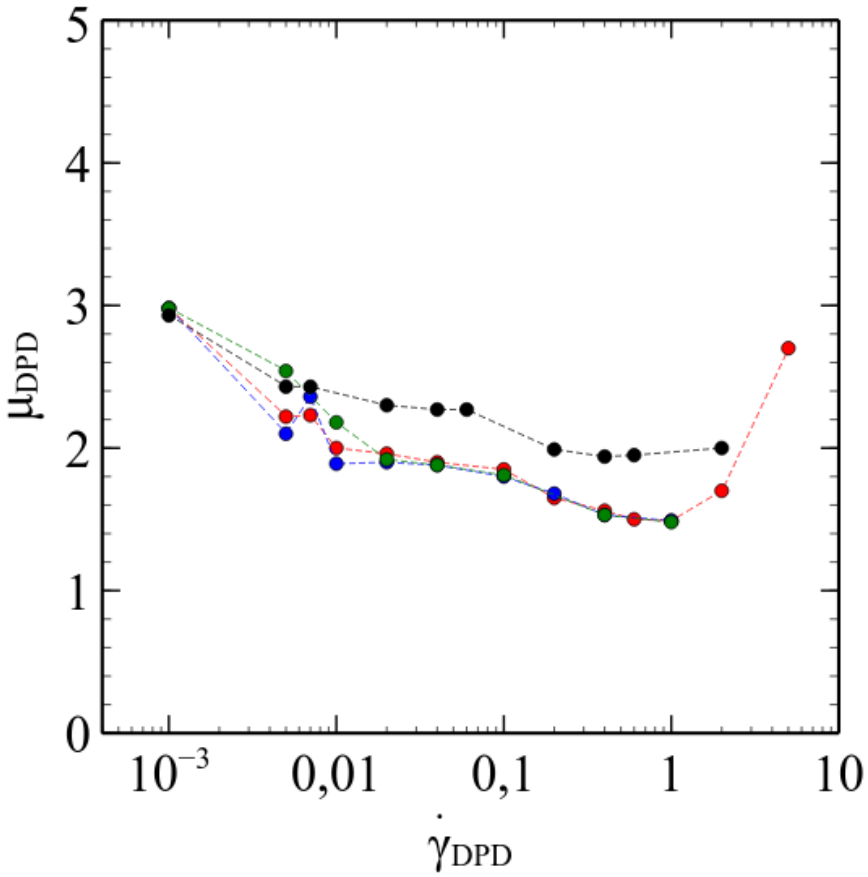


Figure 6.22. *Viscosity, in DPD units, of a mixture of Pluronics L64 in water (25% wt) against shear rate (in DPD units) obtained by varying the harmonic constant (black: 4,0, green: 50.0, blue: 100.0, red: 200.0) in a box with length equals to 30x* r_c

The tuning parameters, obtained from the previous analysis, were transferred to the FENE potential, that reduces the probability of over-extended bonds, due to the presence of a logarithmic term, which produces an infinite energy if particles move too far from each other. Stiffer chains were obtained in this way, and the distance between beads was kept bounded. No significant modifications were observed in the phase diagrams, even if longer equilibration phases were required, due to the reduced mobility of the chains. We also compared the viscosity curves for different concentrations using the two potentials, FENE and Harmonic. We observed, as it is possible to appreciate in Figure 6.23, that a small difference in the numerical value is present, but the shape of the curve is similar.

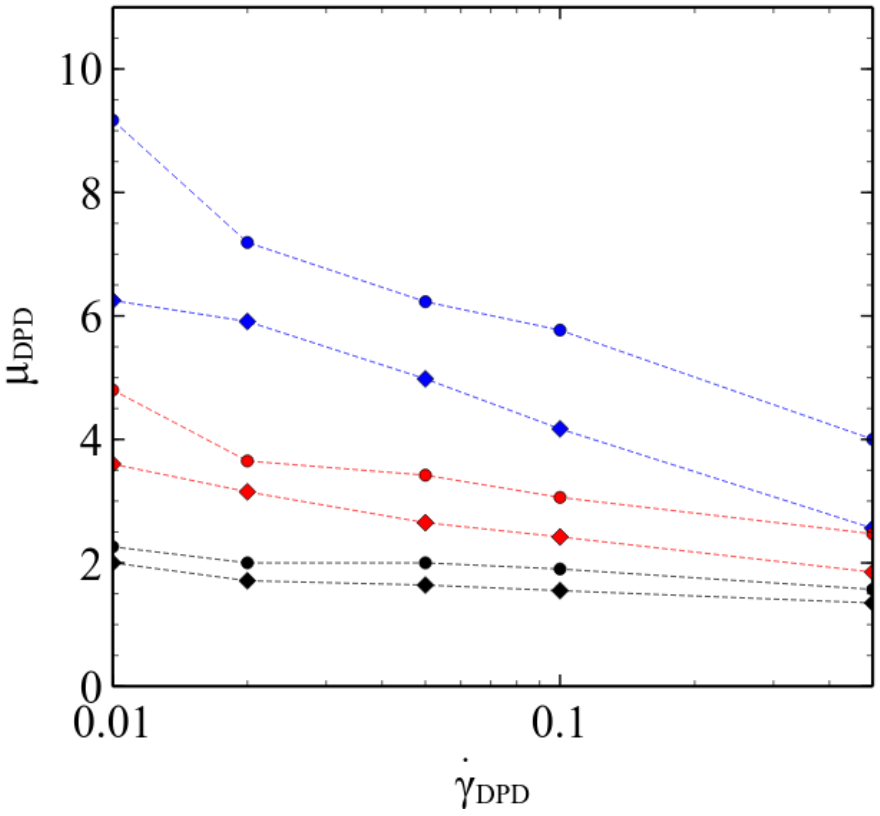


Figure 6.23. *Comparison between the obtained viscosity, in DPD units, against the shear rate, again in DPD units, by using Harmonic (circles) and FENE (diamonds) bonds in the polymeric chains. Colors refer to different concentrations (black: 25% wt, red: 45% wt, and blue: 75% wt) of Pluronics L64 in Water.* κ_{harm} *and* κ_{FENE} *are equal to 50 DPD units.*

Higher differences can be observed at higher concentrations, even if the behaviour of the curve, hence the non-Newtonian behaviour can be appreciated for both the cases. It is important to highlight that at this stage, since a numerical conversion between DPD and physical units is not unique, both results cannot be matched to real examples to validate which case is closer to the reality.

The choice of three different concentrations is related to the different phases that were obtained in the phase diagram. The black curve refers to the micellar region, the red curve to the bi-continuous phase and the blue curve to the lamellar phase. Lamellar phase needed more time to reach equilibrium in case of high shear, because the initially oriented lamellae, destroy their structure to align the sheets to the flow.

Once the system is completely defined, and tuning parameters are selected, we started our analysis to understand how streaming effects were affecting the microstructures that are created at equilibrium, and understanding if these modifications were related to the non-Newtonian behaviour of these complex fluids. The first part of our analysis was performed by re-adopting the clustering algorithm, in

order to evaluate any difference between the equilibrium and non-equilibrium counted structures. We focused our attention on three different concentrations. We selected spherical micelles, gel and lamellae as equilibrium references. The cluster algorithm counted the number of individual structures

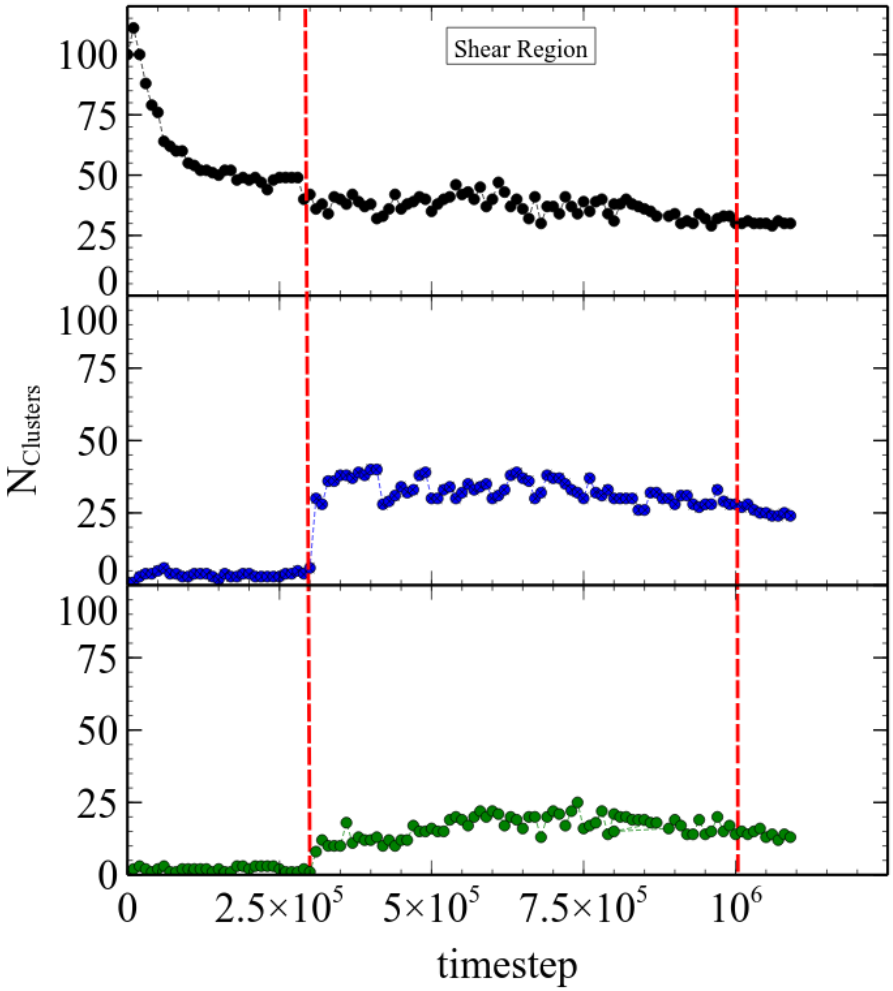


Figure 6.24. *Number of clusters identified in each simulation box as a function of the timesteps: three different concentration of Pluronics L64 (black: 25% wt, blue: 45% wt, green: 75% wt), in equilibrium and non-equilibrium (red-dashed interval) simulations.*

Different aspects can be evinced already at this stage of the analysis. In the micellar region, after an initial equilibration phase, in which the number of spherical aggregates becomes constant, when shear is applied, they start flowing and when colliding, they could end up merging into bigger aggregates. During the *shear phase*, the number of aggregates and their size is affected by a bit of noise related to small elongation of the spherical aggregates that cause the identification of slightly bigger or smaller clusters

(that are considered as new identities). However, it is possible to appreciate that when shear is removed, few bigger aggregates remain into the system. At slightly higher concentration, the number of PPO beads is too high for the clustering algorithm to discern between different structures. Also, it seems that a unique structure composed by all the beads in the system is present at equilibrium. Shear effects promote the formation of new microstructures, clearly distinguished by each other and their number is almost stable during all the shear phase. After the shear is removed, structures are stable for the following timesteps. Finally, at very high concentration, the situation is similar to the intermediate one. The number of structures cannot be identified, but during the shear phase, varies usually between 5 and 10 structures. In this last case, the system explores the lamellar phase, that can be recognized from the previous analysis in the equilibrium structure, but a more accurate analysis in the interior structure, showed that interconnections between lamellae are present. These interconnections do not allow the cluster algorithm to recognize separate lamellae during the equilibrium phase. Instead, the application of shear rate on the system, destroy these connections, by aligning lamellae to the flow and clearly separates the single sheets.

An interesting aspect can be appreciated at intermediate concentrations. For example, if we look at the case where the concentration of Pluronic L64 is around 60%. We can observe in Figure 6.25 (left), that an interconnected gel is formed at equilibrium. Interconnected elongated structures come in contact, creating a disordered solid structure. However, the application of the shear on the system destroys the network to create an hexagonal phase in which cylinders, aligned to the flow field, are separated by fixed distance. This aspect has been also highlighted in the cluster analysis, for intermediate concentrations. This specific case can be used as proof that streaming effect can modify equilibrium configurations and produce modifications in the microstructure of a complex fluid. In Figure 6.25 (right), it is possible to appreciate the interconnected network, composed by disordered linked structures, is deformed when shear is applied. Hexagonal microphase can be observed in a frontal plane, perpendicular to the flow plane. These cylindrical structures, composed by PPO beads in the core, and PEO (removed from the picture for sake of clarity) in the exterior part, extend along the direction of the flow. Cylinders can be easily counted by using the clustering algorithm. Similar structures can be also found if the shear rate is slightly increased or decreased. However, when it crosses certain limit value, i.e. more than 2 DPD units, these structures are mostly flushed away by the streaming effect.

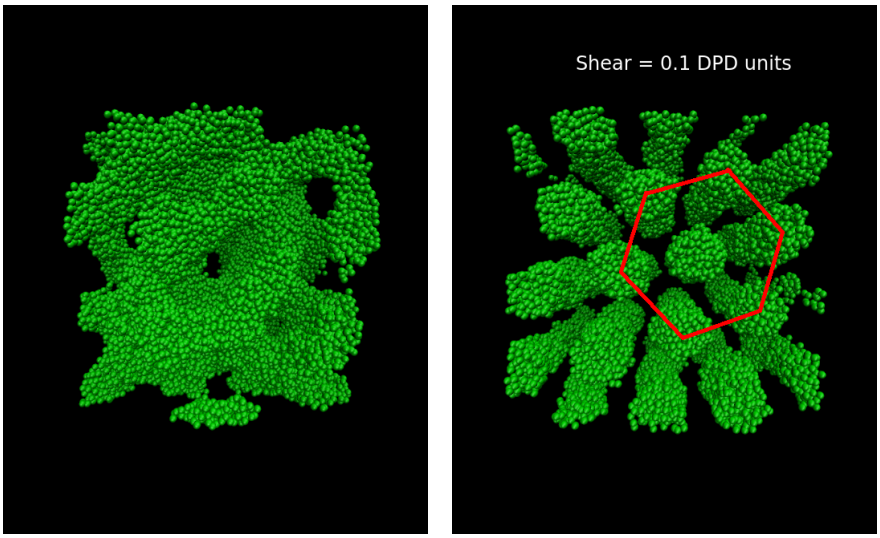


Figure 6.25. *Morphological transition, obtained at 60% wt of Pluronics L64 in water, from a disordered interconnected structure into a more ordered hexagonal phase when a uniform shear of 0.01 DPD units is applied on the simulation box. Hexagons that are formed can be easily counted.*

Similar tests were performed at this stage by using another simulation code. To validate this modification of the microstructure due to the flow effects and the deviation from the equilibrium configuration, we run similar cases on DL_MESO. We kept the same concentration, shear, and LEBC since already implemented into DL_MESO. There is a slight difference between DL_MESO and LAMMPS in reproducing LEBC. While in LAMMPS, the velocity is zero at the bottom of the box and maximum at the top of box, in DL_MESO the velocity is zero at the center of the box and the velocity is half of the maximum value at the top and half at the bottom but in the opposite direction. A linear velocity profile, hence a constant shear stress can be obtained in both cases. As it is possible to observe in Figure 6.26, in the top part, equilibrium configuration is represented by an interconnected network and again only the PPO beads are showed. In the bottom part, a shear rate of 0.1 DPD units was applied on the box and the cylindrical structures, oriented in an hexagonal pattern were found. For the sake of clarity, the structures are less dense, because at this stage we tested a smaller box on DL_MESO only to prove that similar patterns were recognized. The box reported in Figure 6.27 has a length size of $20 \times r_c$ and a DPD density equals to 3.0.

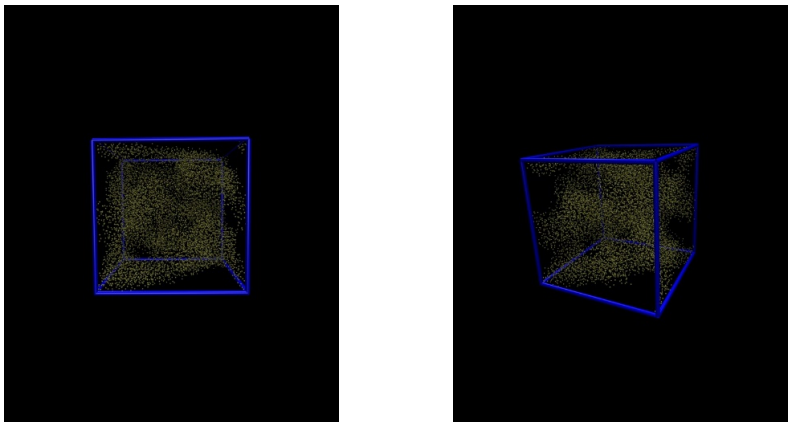


Figure 6.26. *Equilibrium configuration of a mixture of Pluronics L64 in water obtained with DLMeso, by using the interaction coefficients already used for LAMMPS simulations. The two snapshots represent two different views of the same system*

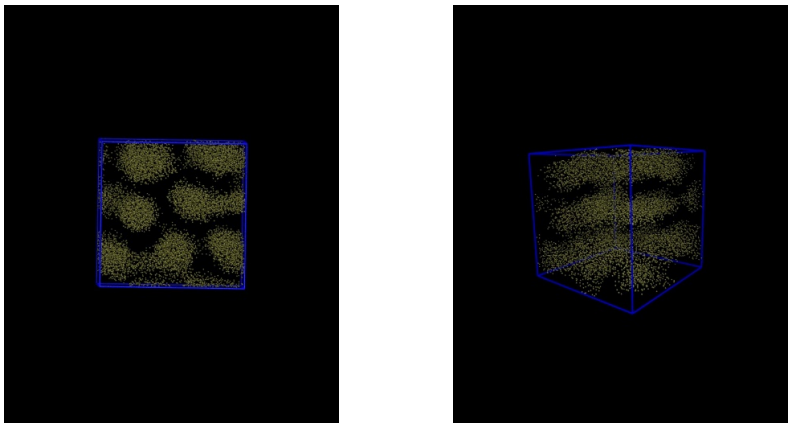


Figure 6.27. *Non-Equilibrium configuration of a mixture of Pluronics L64 in water obtained with DLMeso, by using the interaction coefficients already used for LAMMPS simulations. Again, the morphological transition from a disordered into hexagonal phase can be appreciated with this simulation software.*

The final part of our analysis using DPD, after the validation of the phase-diagrams of two different systems, is focused on the rheological response of these system to shear stress. We have already discussed and validated the clustering algorithm and in Figure 6.28, it is possible to appreciate how the main structures highlighted in the previous analysis for the system composed by Pluronic L64 into water, change when shear is applied. It is possible to observe that in the cases of middle and high concentration (mid and right), the cluster algorithm is not capable of distinguishing single structures at equilibrium, which translates into a unique red blob. It must be highlighted that in the case of high concentrtion, microstructure is lamellar, before and after the application

of the shear rate. However, the interconnections between lamellae do not allow the algorithm to separate the lamellae. When shear is applied, the separation is neat and structures can be easily counted.

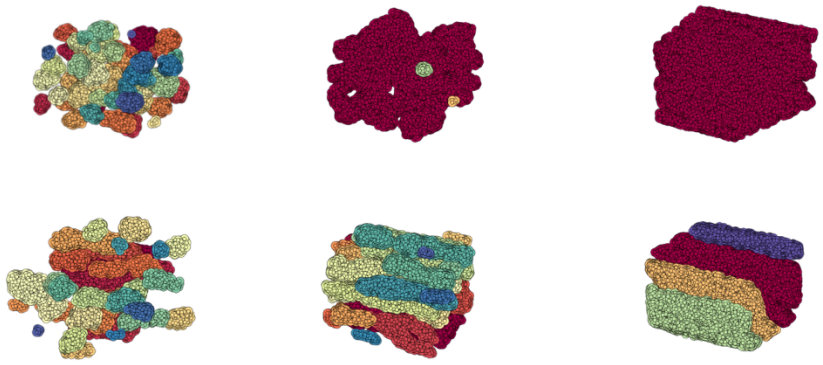


Figure 6.28. *Snapshots of three different concentrations (left: 25% wt, middle: 45% wt, and right: 75% wt) of Pluronics L64 in water analyzed with the clustering algorithm. Colors refer to single structures identified by the algorithm.*

For example, spherical and elongated micelles, move under streaming flow but they do not significantly change their shape. During the shear event, micelles may coalesce because they come closer, but they can also be destroyed. Differences can be appreciated in the intermediate case, where the transition from a soft-gel to oriented hexagonal structures. Cylinders are oriented in the direction of the flow and in a frontal plane they appear to be located in a fixed pattern as previously showed. The same concepts apply to the highest concentration. Actually, the continuous structure, that has been identified as a unique red blob in the equilibrium configuration by the cluster algorithm, is composed by lamellae but interconnections between the single sheets, do not allow the single structures to be identified. The separation becomes clear when shear is applied. Lamellae are destroyed and re-oriented to follow the streaming effect and the single sheets become easier to identify. Since this analysis was only qualitative, we also tried to quantify how the shear was affecting the equilibrium structure from a quantitative point of view. Thanks to the clustering algorithm, we were able to obtain an averaged cluster mass distribution for all these cases, before and after the shear. Three different concentrations have been tested and the average cluster mass distributions (CMD) are reported in Figs 6.29-6.31

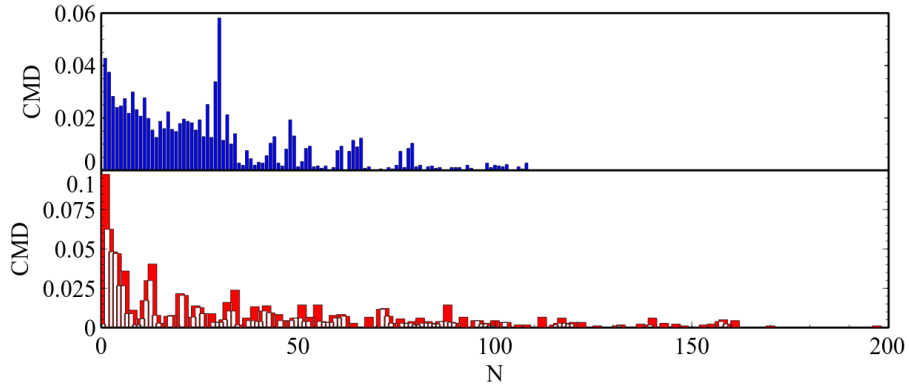


Figure 6.29. *Cluster mass distribution (CMD), i.e. the probability of finding a cluster of dimension N (number of chains in one structure) in the simulation box, plotted against N, for a concetration of 25% Pluronics L64 in water at Equilibrium (blue) and Non-Equilibrium (red). Shear rate is equal to 0.01 DPD units. White histograms are a graphical artifact to show close values overlapping.*

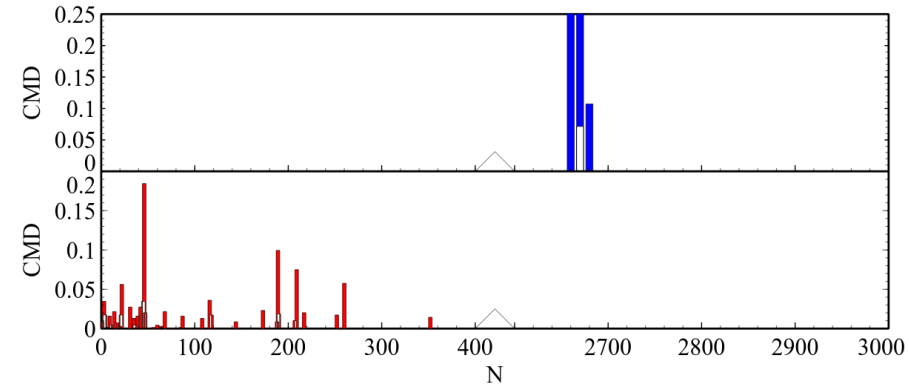


Figure 6.30. *Cluster mass distribution (CMD), i.e. the probability of finding a cluster of dimension N (number of chains in one structure) in the simulation box, plotted against N, for a concetration of 45% Pluronics L64 in water at Equilibrium (blue) and Non-Equilibrium (red). Shear rate is equal to 0.01 DPD units- White histograms are a graphical artifact to show close values overlapping.*

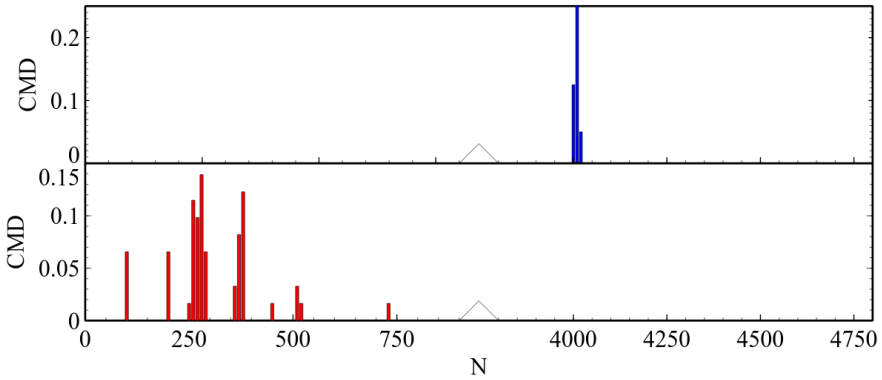


Figure 6.31. *Cluster mass distribution (CMD), i.e. the probability of finding a cluster of dimension N (number of chains in one structure) in the simulation box, plotted against N, for a concetration of 75% Pluronics L64 in water at Equilibrium (blue) and Non-Equilibrium (red). Shear rate is equal to 0.01 DPD units. White histograms are a graphical artifact to show close values overlapping.*

It is possible that in the case of low concentration of Pluronic L64, the distribution slightly changes, because of aggregation phenomena that may happen, through the creation of few new clusters bigger than the original structures. Even if these structures are bigger, their shape does not change as it can be observed in Figure 6.32, where the analysis of the gyration radius is reported. It is possible to see for two different concentrations, 5% w/w and 25% w/w that when shear is applied, the gyration radius and the number of chains per aggregate increase, but the exponent previously discussed does not change. This ensures that micelles move because of the streaming effect, they could coalesce, but never deform.

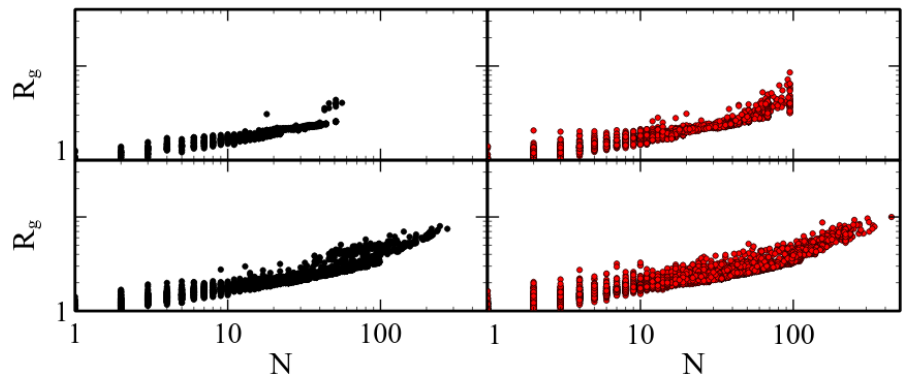


Figure 6.32. *Comparison between the gyration radius reported for two concentrations (5% (black) and 25%(red) wt) in equilibrium (top) and Non-Equilibrum (bottom) configurations. It is possible to appreciate that all the points lie on a line with the same slope, except for few bigger aggregates that result from the coalescence of smaller ones. This explains that the shear has no effect on the morphology of such systems that keep their spherical shape.*

If we look now at the intermediate concentration, we can see that the uniform blob that mainly contains all the PPO beads, breaks into smaller aggregates. These new aggregates can have different size. Mostly, they are cylindrical structures with at least 200 beads, but also elongated micelles, with fewer elements outnumber the cylinders. Even if the overall structure is ordered, the dimension of each cylinder is variable also because of the confined bulk system and the periodic boundary conditions, that may cause a unique cylinder to be counted as two different elements. Finally, in the last case, the blob splits into lamellae. In this case, less structures are identified for each timestep, so the distribution is quite scattered. However, it is clear that the system is composed by elongated sheets, fewer in number compared to the cylindrical structures, that contain more elements. In this last case, periodic boundary conditions were not adopted, because of the high number of PPO beads, too many to be handled without parallelization.

A final validation can be used to validate the modification of the microstructures obtained in complex fluids. Thanks to experimental data for the system composed by Pluronic L64 and water on the viscosity curves, we were able to verify the behaviour of such fluids. In fluids composed by different components, distinct structures can produce peculiar rheological behaviour. Non-Newtonian phenomena, for example, can be observed when the viscosity of a system drops or increases when the fluid undergoes shear stresses. When the viscosity of a fluid increases with the shear (shear-thickening behaviour) we talk about dilatant fluid, while when the shear has the opposite effect (shear-thinning behaviour) we call it pseudo-plastic. Even if a direct comparison cannot be made in terms of units, it is possible to compare the behaviour of these systems. In Figure 6.33 and Figure 6.34 are reported simulation and experimental results of Pluronic L64/water mixture. The two curves are obtained with two different

experimental techniques that are able to reproduce a steady shear condition that can be compared with DPD equilibrium simulations, and a dynamic condition that can be compared with DPD non-equilibrium simulations. It is possible to observe from these curves that in the range of low concentration the two curves match, meaning that shear is not affecting the structures, as it is also possible to observe in DPD simulations. However, when the concentration is increased the two curves diverge. This is a clear signal of differences in the microstructures that are contained in the system. In DPD simulations this effect can be appreciated by a drop in the viscosity due to flowing, merging and aligning effects of the microstructures

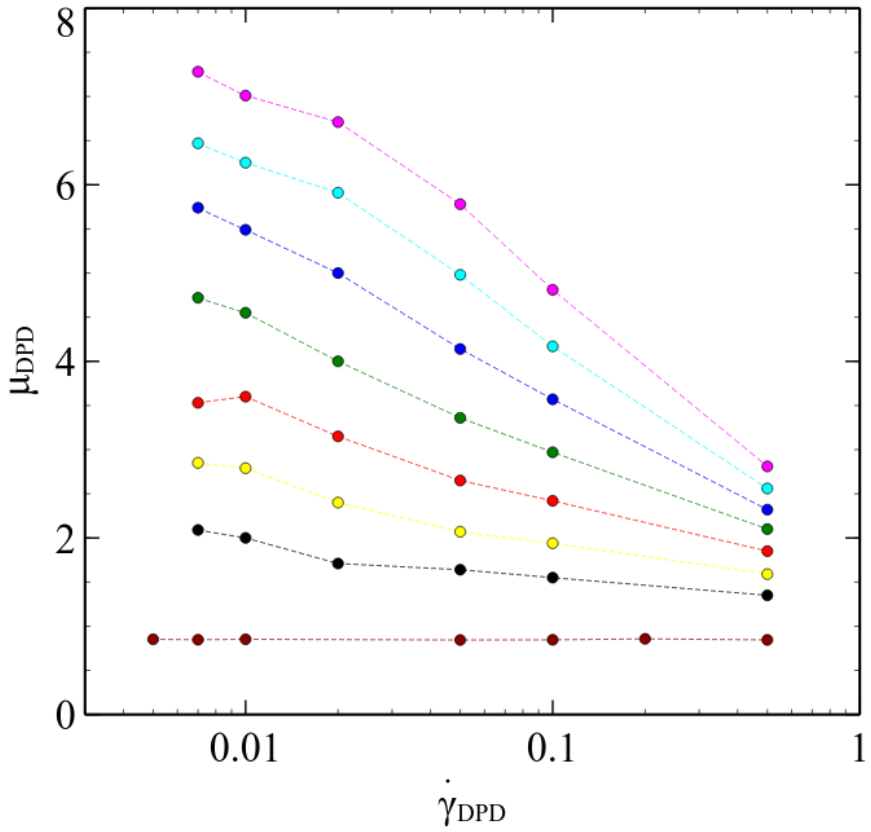


Figure 6.33. *Viscosity, in DPD units, reported against the shear rate, again in DPD units, for different concentrations (amaranth:* 0%*, black:* 25%*, yellow:* 35%*, red:* 45%*, green:* 55%*, dark blue:* 65%*, light blue:* 75%*, purple:* 85% *wt) of Pluronics L64 in water with FENE potential.*

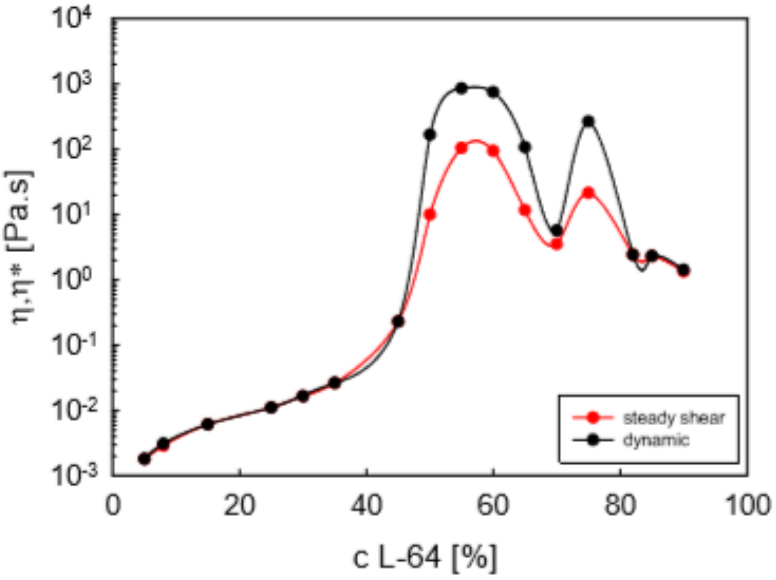


Figure 6.34. *Experimental steady shear and dynamic viscosity of a mixture water/Pluronics L64 at different concentration.(Pasquino et al., 2019)*

Curves, representative of increasing concentration of Pluronic L64, are compared with pure water (amaranth). We omitted low concentration curves (i.e. less than 25%) because they were all flat curves. The first curve, which is still in the Newtonian regime, is the 25% of concentration. In order to appreciate a drop of the viscosity, we must move to intermediate concentrations (i.e. more than 45%), where elongated and spherical micelles are replaced by interconnected network. In this case, the initial viscosity drops while shear is increasing, meaning that microstructure of the system is changing, and disordered structures are becoming more ordered and following the streaming.

A final remark is necessary. DPD models have been used so far to obtain qualitative and quantitative analysis at equilibrium, and mostly qualitative predictions in non-equilibrium. A standardized conversion rule between DPD and real units has not been provided yet. This operation can be easily obtained at equilibrium, because the conversion is based on the number of beads clustered into one bead and on their thermal velocity. This approximation may lead to erroneous results if the same conversion set is used for non-equilibrium simulations. Using an approach that is similar to the one proposed by Groot and Rabone, the shear values in real units obtained in the range of analysis are extremely high. The order of magnitude of the shear obtained in this way, falls between 10^{10} and $10^{11} s^{-1}$, which is not comparable with the shear rate obtained by experiments.

6.3 Emulsions

In this section, a system composed by water and silicone-oil emulsion, used in personal care product manufacturing is investigated. The formula is kept unknown and the different silicone-oils are identified by their different viscosities. Two geometries are explored, a lab scale mixer and a pilot plant mixer that are used for similar purposes.

6.3.1 ESCO 6L

ESCO 6L mixer has been simulated using Ansys Fluent. In the specific case, different mixtures of water and silicone oil were tested. The differences between silicone oils are related to the viscosity of such systems. The analysis was initially focused on the prediction of the fluid dynamics of the system. Two main batches of simulation have been initially tested, including and removing the anchor. In the experimental setup, this element is fixed and it acts only as a baffle by modifying the flow field developed. We tested both the system with and without the anchor. According to the experimental setup, all the relevant fixed parameters are reported in Table 6.3.

Table 6.3: ESCO 6L parameters of the laboratory equipment.

Parameters	**Value**
Anchor, fix/rot	fix
Impeller Diameter, m	0.05
Velocity, RPM	3000
Velocity, rev/sec	50.0
Phases	Water/Silicone oil
Density	$1000 kg/m^3$
Constructor Power Number	0.30
Tank Radius, m	0.10
Tank Height, m	0.30
Tank Volume, l	9.40
Tank Operating Volume, l	6.00
Power Draw, W	12.90

CFD models were validated against given value of the constructor power number (Gao et al., 2016; D. Li et al., 2017). The power number (PO) of an impeller can be calculated according to two different equations, based on the torque (T_q) acting on the rotating and fixed parts of the system, or on the dissipation rate (ϵ) of the turbulent kinetic energy, which is an indicator of the dissipated power:

$$PO_{CFD} = \frac{2\pi\frac{N}{60}T_q}{\rho_f\left(\frac{N}{60}\right)^3 D^5}, \tag{6.3}$$

$$PO_{CFD,diss} = \frac{<\epsilon> V_f}{\left(\frac{N}{60}\right)^3 D^5}, \tag{6.4}$$

where N is the rotation speed of the impeller, V_f is the volume of the tank, ρ_f is the density of the fluid and D is the diameter of the impeller. In many cases, these two numbers are not equal in CFD calculation. We initially tested the effect of different meshes on the power number, in order to obtain mesh independence results, as it is possible to observe in Table 6.4:

Table 6.4: Meshes and operating conditions tested for the ESCO 6L in Ansys Fluent. Power number obtained from the torque and the turbulent dissipation rate is compared against the constructor power number.

Testcase	PO_{CFD}	PO_{diss}
Experimental 3000RPM (Target)	0.32	0.00
Experimental 2000RPM (Target)	0.32	0.00
Tetrahedrons, 2000RPM	0.90	0.91
Tetrahedrons, 2000RPM, Refined	1.25	1.12
Tetrahedrons, 2000RPM, Refined x2	1.08	1.03
Tetrahedrons, 3000RPM	1.09	0.78
Hexahedrons, 2000RPM	1.30	1.00
Hexahedrons, 2000RPM, Refined x2	1.27	0.95
Hexahedrons, 2000RPM, 6L	1.09	0.36
Hexahedrons, 2000RPM, 6L, Refined	1.09	0.33
Hexahedrons, 2000RPM, 6L, 2Phases	1.45	1.42
Hexahedrons, 2000RPM, 6L, 2Phases, Refined x2	1.20	0.88
Hexahedrons, 3000RPM, 6L, 2Phases	1.27	0.91
Hexahedrons, 3000RPM, Reynolds Stress	1.27	0.41
Hexahedrons, 3000RPM, Reynolds Stress, Refined	0.99	0.41
Hexahedrons, 3000RPM, Reynolds Stress, Refined x2	0.99	0.41

Two topologies of mesh were tested, tetrahedrons and hexahedrons dominant patterns. The geometry of the experimental mixer has a working capacity of around ten liters, but simulation and experiments were performed in the half-filled test case. Initial validation was aimed at finding the minimum number of cells, hence the minimum resolution, that provides grid independence. We used the constructor power number as reference, even if small deviations may be caused by the operative condition. If the constructor power number is obtained with the impeller in straight position and in a tank with no obstacles, we computed the simulations in the experimental setup. In this particular case, the impeller is inclined by a certain angle and an anchor is present. Differences may be related to this aspect, but we noticed that there was no big variation by increasing the resolution of the mesh. In particular, we saw that regular hexahedrons were better performing in all the cases, and in both cases the region around the inclined impeller had to be refined. The number of elements was increased close to impeller and anchor zones, but also in the interfaces between rotating and static region. If the resolution of the mesh is insufficient in these areas, discontinuities in the velocity field are obtained, due to defects in the interpolation between communicating cells. The effect of motion of the impeller is caused by applying rotating cell conditions in the volume surrounding the impeller, which instead is fixed in its system of reference. For the hexagonal case, when the mesh was composed by more than 1 000 000 elements, no improvement were noticed. We tested meshes up to 4 000 000 elements, in steady state simulations, for the development of the velocity field. We fully developed the velocity field starting with 100 RPM, and increasing to respectively 200, 400, 800, 1600, 3000 RPM. In the first batch of simulations, turbulence model was off, while we switched it on when we crossed the 1000 RPM. Between 2000 and 3000 RPM, the turbulence is fully developed. At this stage, we also tested different turbulence model. in particular we focused our attention on RANS (Reynolds Averaged Navier-Stokes) models and avoided Large Eddies Simulations (LES). $\kappa-\epsilon$ (e.g. standard, RNG, realizable),$\kappa-\omega$-SST, and Reynolds stress models were tested and compared for both the systems (i.e. with and without anchor), and a summary of the most relevant results is reported in Table 6.5:

Table 6.5: Turbulence models and drag-forces tested for the ESCO 6L in Ansys Fluent, for the same mesh (Hexahedrons, 3000 RPM, Refined). Power number obtained from torque and turbulent dissipation rate are compared between them.

Testcase	**Tr, Nm**	$PO_{CFD,impeller}$	$PO_{CFD,wall}$	$\epsilon, W\backslash kg$	$PO_{CFD,diss}$
$\kappa - \epsilon$	0.072	1.28	1.30	1.89	0.930
$\kappa - \epsilon - RNG$	0.076	1.34	1.38	0.76	0.372
$\kappa - \epsilon - realizable$	0.071	1.32	1.28	1.81	0.892
$\kappa - \omega$	0.071	1.34	1.28	1.45	0.715
Schiller-Naumann	0.073	1.43	1.32	0.90	0.440
Morsi-Alexander	0.073	1.43	1.32	0.90	0.440

In Tab 6.5, it is also possible to find the results obtained by introducing the disperse phase, i.e. low viscosity silicone-oil. We tested different drag models, since this force has to be properly modeled. No differences were found between Schiller-Naumann and Morsi – Alexander, while we avoided other models implemented in Fluent, since they are more suitable for gas-liquid or solid-liquid systems. Schiller-Naumann was selected for all the remaining simulations. After identifying the models needed to describe the fluid dynamics of the mixer, we obtained the fully developed velocity field for the system containing both continuous and disperse phase (1% of silicone-oil in water). In Figures 6.35 and 6.36, it is possible to appreciate, for two different impeller speeds (2000 and 3000 RPM), velocity vectors representing the flow around the impellers. Highest velocity is obtained at the tips of the impeller, slowly decreasing while moving apart from the rotating region. In general, the impeller causes turbulent flow, mostly in its surroundings, while other regions of the tank do not experience such flow. It is important to underline and figure out this aspect, because the region closer to the impeller is the one where the value of the turbulent dissipation energy is higher and also in the experimental setup, a sampling pipe is located in this region.

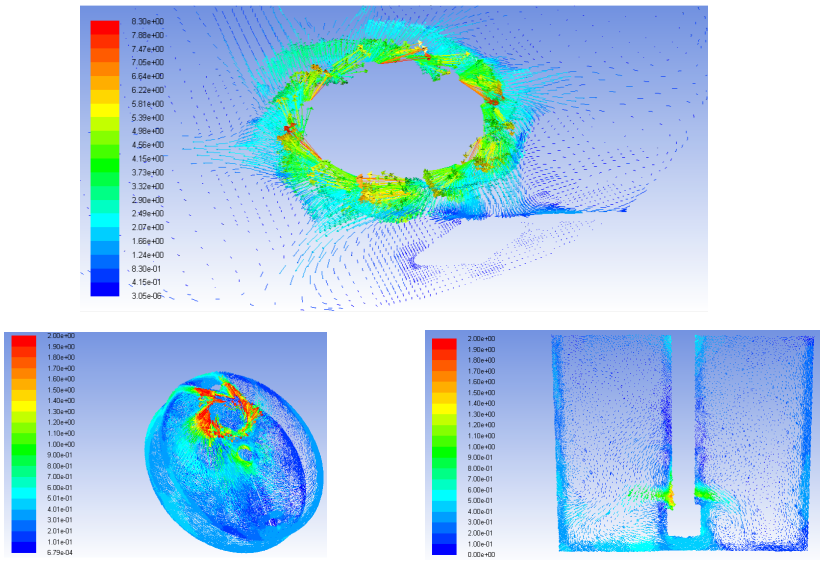


Figure 6.35. *Snapshots of the velocity vectors and contours obtained around the inclined impeller for a speed of rotation of 3000 RPM. Different snapshots refer to different planes. It is possible to appreciate the evolution of the velocity field around the impeller and the modification due to the presence of the anchor.*

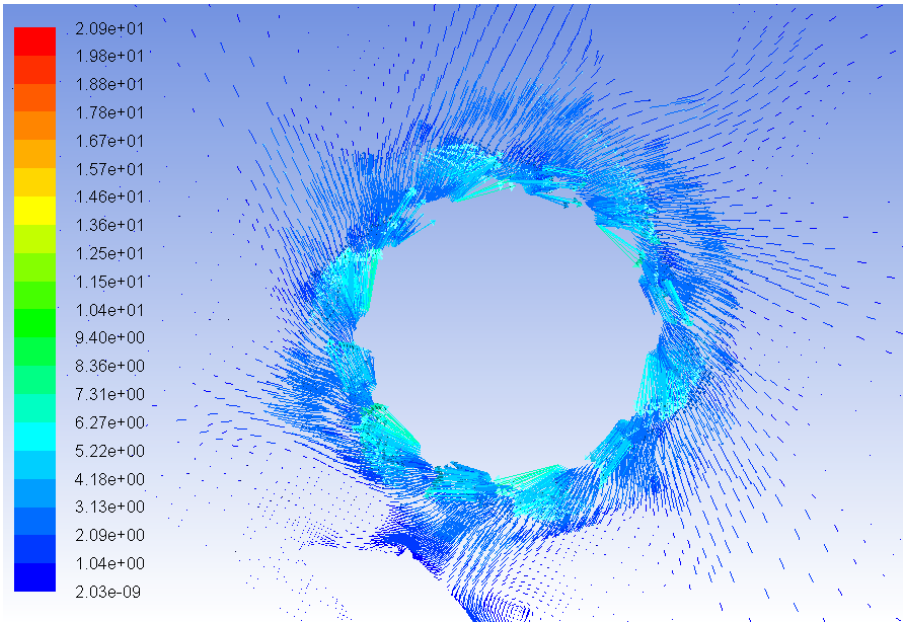


Figure 6.36. *Snapshot reporting the velocity vectors obtained around the inclined impeller for a speed of rotation of 2000RPM. The pattern is similar to 3000RPM because we are in fully turbulent regime in both cases.*

A summary of this analysis can be found in Figure 6.37 and Figure 6.38, where

for two different turbulence models, a total of 12 meshes hexahedral and tetrahedral dominant, with increasing refinement, were tested. The value of the power number obtained by the torque is an average between the torque applied on the rotating part and the one applied on the fixed walls. These two values were, in most of the cases, equal. This means that the same force that is exerted by the impeller on the surrounding fluid, is used to keep fixed the walls of the tank.

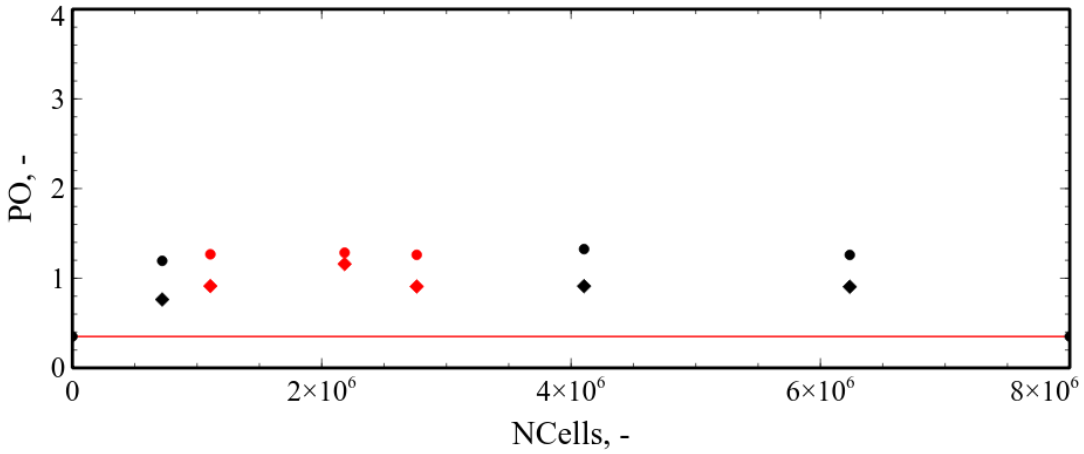


Figure 6.37. *Mesh independence study validated by comparing the power numbers calculated from the torque acting on the impeller (circle) and the turbulent dissipation rate (diamond) for hexahedrons dominant (red) and tetrahedrons dominant (black) grids with $\kappa - \epsilon$ turbulence model.*

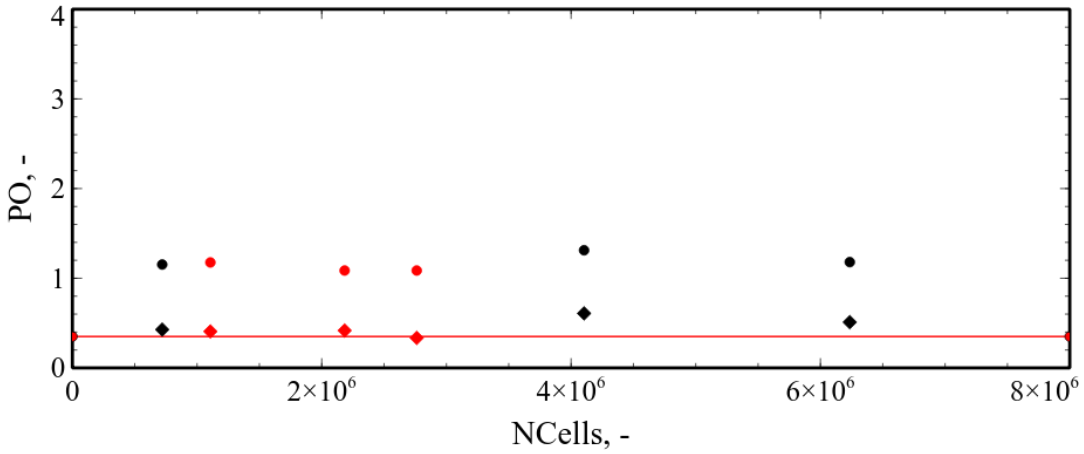


Figure 6.38. *Mesh independence study validated by comparing the power numbers calculated from the torque acting on the impeller (circle) and the turbulent dissipation rate (diamond) for hexahedrons dominant (red) and tetrahedrons dominant (black) grids with Reynolds stress turbulence model.*

From these figures, it is possible to appreciate that the power number obtained from the torque is comparable with both the turbulence models, and it does not change significantly with the number of mesh elements. Instead, the power number obtained through the turbulent dissipation energy is slightly different and Reynolds-stresses model, where all the component of the stress tensor are solved, are in better agreement with the constructor power number. Errors of around 5% were obtained in this last case, while three times higher power number was obtained using the standard $\kappa - \epsilon$ model. At this stage, we had a complete overview on mesh, models and numerical details to run simulation with population balance equation. In Figure 6.39 and Figure 6.40, are reported the fully developed velocity fields for two case, with and without anchor, with $\kappa - \epsilon$ and Reynolds stress model. It is evident that turbulent dissipation rate is higher in the region closer to the impeller. In general, a velocity field is developed in all the volume and vortices can be identified. However, in some areas, the velocity of the fluid is almost zero, meaning that stagnant zone are also present, mostly in the areas far from the impeller and anchor or on the top of the tank.

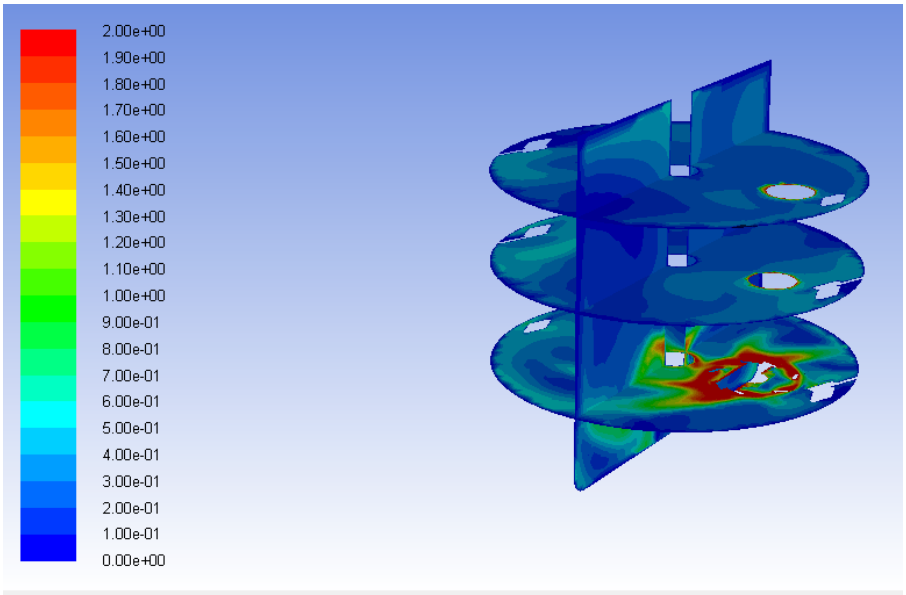


Figure 6.39. *Countour plot of the velocity field obtained including the fixed anchor, with $\kappa - \epsilon$ turbulence model, at 3000 RPM. The grid is composed by 2 000 000 hexahedrons, refined around the rotating region and the anchor.*

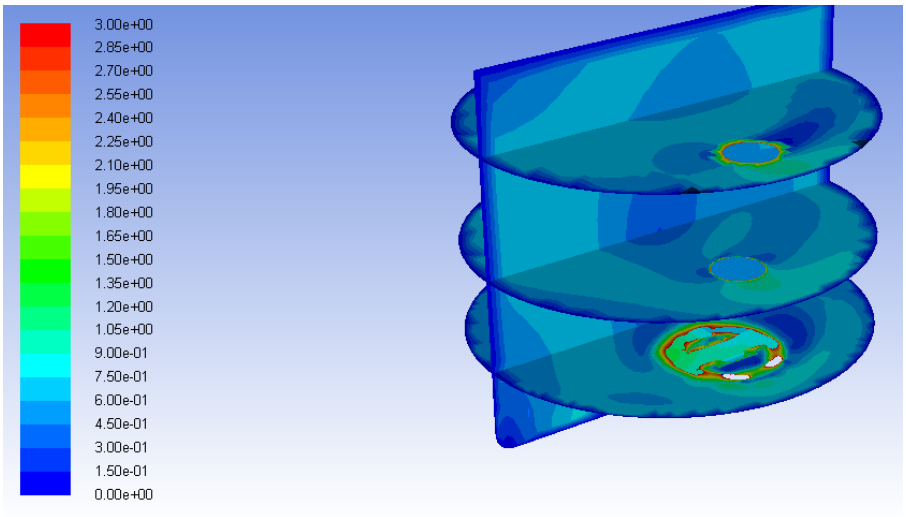


Figure 6.40. *Countour plot of the velocity field obtained including the fixed anchor, with Reynolds stress turbulence model, at 3000 RPM. The grid is composed by 800 000 hexahedrons, refined only around the rotating region.*

After we obtained a fully developed velocity field, for a system composed by water, 1% of silicone-oil, and surfactant (SLES), we tested if by using the population balance equation and the QMOM to solve the equations, by varying the kernels, we were able to track the evolution of the droplet size distribution within the mixing tank. We tested different kernels (i.e. Laakkonen-Alopaeus, and Coulaloglu-Tavlarides), with four silicone-oils (from 0.5mPas to 242 mPas) at different viscosities, using two computational codes (i.e. MATLAB and fluent). We compared these results with experimental data obtained by El Hamouz (EL-Hamouz et al., 2009) in the ESCO 6L mixer. Experimental data describing the droplet size distribution were obtained for different silicone-oil in water mixtures with increasing viscosities. The addition of small amount of SLES and the reduced size of the droplets prevent coalescence phenomena. In the experiment, the oil is introduced while the anchor is rotating. When the concentration can be considered uniformly distributed in the mixing tank, the anchor is stopped while only the impeller is turned on. We simulated this second part of the experiment, where we assumed that the secondary phase (silicone-oil) is already introduced and homogeneously dispersed into the system, by patching the value of the disperse phase on the whole domain in Fluent. In Figure 6.41, the experimental values are reported. It is possible to see that the mixing is stopped after 200 min even if stable mixtures end after around 50 minutes. This was done to guarantee the absence of any coalescence phenomena and the stability of the mixtures. Also, the final products were collected and put to rest, and the measurements were repeated after few days, but no changes were recorded in the equilibrium droplet diameters.

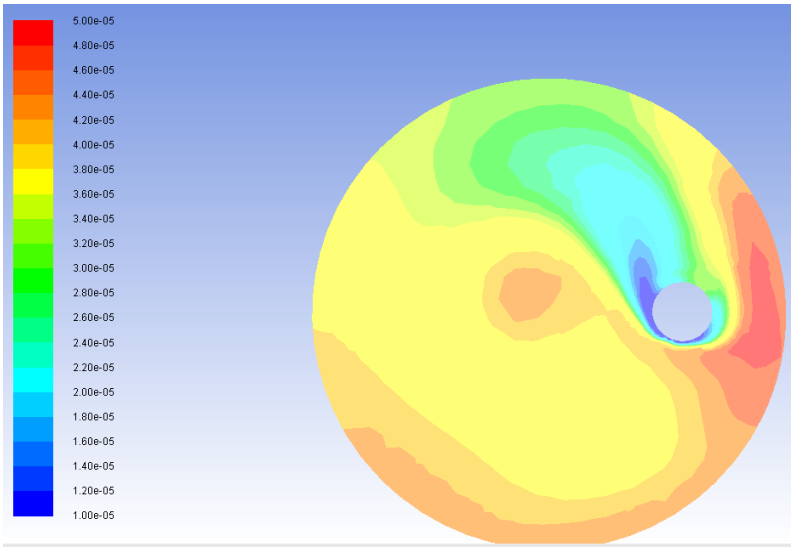


Figure 6.41. *Example of the Sauter diameters distribution on the plane of the impeller, using Tavlarides kernel, after 600s and starting with an initial diameter of 5.5e-5 m, at 3000 RPM for a low viscosity oil. Droplets are smaller in the region closer to the impeller where the turbulent dissipation rate is higher compared to the remaining volume.*

In Figure 6.41, an example of contour plot of the Sauter diameter on a plane above the impeller can be observed.

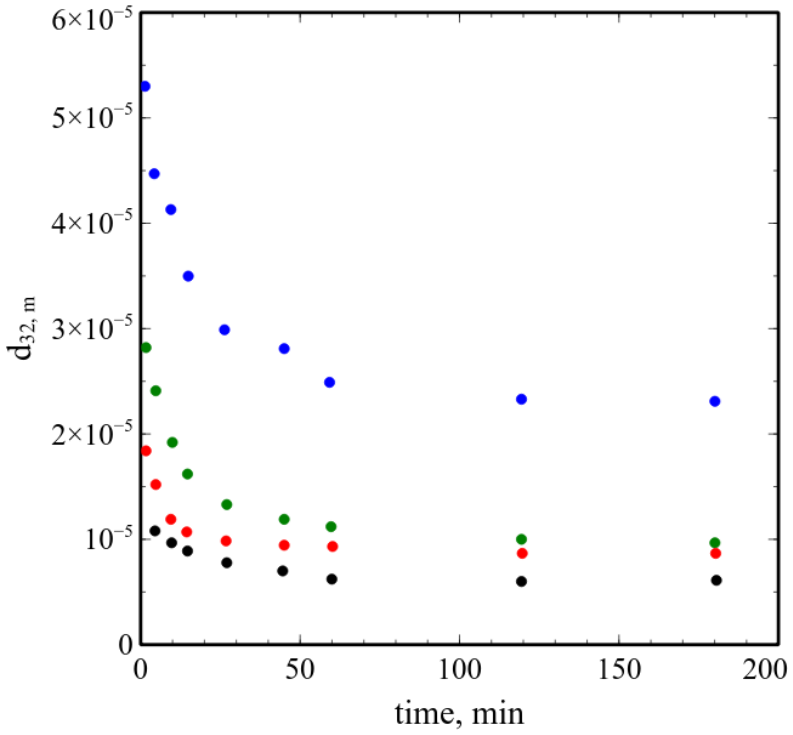


Figure 6.42. *Experimental values of the Sauter diameter for different viscosities (black: 0.5 mPas, red: 12 mPas, green: 30mPas, blue: 242 mPas) of silicone oil in water reported against the stirring time (EL-Hamouz et al., 2009).*

It is clear that smaller droplets are generated in the area where the turbulent dissipation energy is greater, while bigger stable droplets can be observed where the flow is stagnant. The breakup process was simulated for at least 60 minutes, using a timestep of 1 second, and in this time span, the breakage rate dropped to zero for all the cases. This happens because when the droplets become too small, they enter in the viscous sub-range while all the tested kernels are non-zero only in the inertial subrange. The limit between inertial and viscous sub-range can be identified by calculating the Kolmogorov length:

$$L_k = \frac{\nu_c}{\epsilon_{turb}} \tag{6.5}$$

Where ν_k is the kinematic viscosity of the continuous phase, and ϵ is the turbulent dissipation rate, and this value can be very different in the cells of the mixer, because of the wide distribution of the values of epsilon. Using the developed velocity fields for the different oils as starting point, we tested the different kernels. Since the volume fraction of the disperse phase is low, we decoupled the resolution of the flow field from the PBE. This means that PBE was solved without updating the velocity field at every timesteps. However, we updated the velocity field after 150 seconds in every

simulations.We proved that almost no differences were reported if fully coupled solution is compared to the segregated one, but results were omitted for the sake of brevity. The decoupling was possible for this specific case, because the volume fraction of the disperse phase is too low to influence the velocity field in the mixer. When the concentration of the disperse phase is higher, and the viscosity of the system is affected by the DSD, it is necessary to use a fully coupled approach, that will provide more accurate results, but it will cost more computational time.

Lakkonen Kernel

We started our analysis with our own implementation of both kernels and QMOM in Fluent. We tested Laakkonen kernel using constants that we found in the literature, already tested on silicone-oil in water mixtures, but in different mixers. At this stage, we tested two different approaches. Since the volume fraction of the disperse phase was small, and the decoupling possible, we extracted the distribution of turbulent dissipation rate from each cell in Fluent, and run MATLAB simulations of a single cell in time, as proposed by Buffo et al., 2016. The results of this analysis was the evolution of the mean diameter by using the volume average of the turbulent dissipation energy, obtained as:

$$\epsilon_{av} = \frac{\sum_i \epsilon_i V_i}{\sum_i V_i}, \tag{6.6}$$

where ϵ_i is the value of the turbulent dissipation energy in one cell i, V_i and is the volume of the cell. Thanks to this analysis we had preliminary results that were compared to the experimental data. As it is reported in Figure 6.43, we had to test different values of the constant C_3, for the Laakkonen kernel. In fact, the problem we observed was that the proposed constants, were failing in the prediction of the high viscosity curve. This result is reasonable because the constants obtained from the literature were mostly tested on low viscosity systems, and as we can se from our MATLAB simulations, the match between the original constants and the low viscosity oil is almost perfect. If we change the value of the C_3 constant, we can observe that all the curves are shifted down when the value of this constant is reduced. A smaller value of this constant is reflected into a reduced resistance to breakage because of viscous effects that have to be overcome.

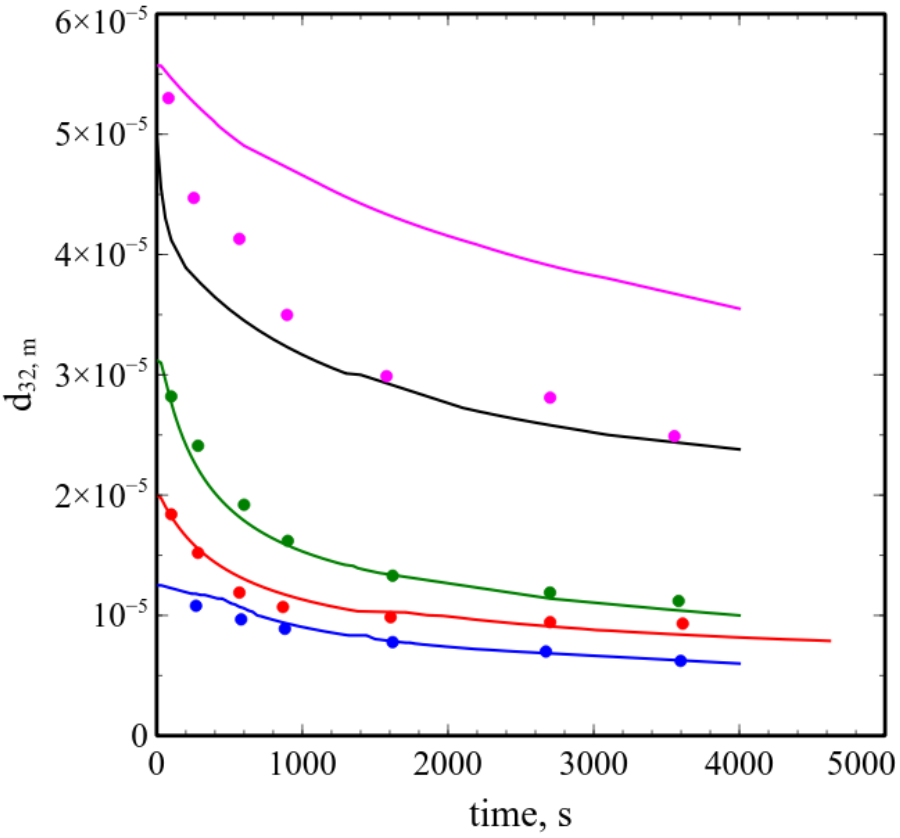


Figure 6.43. *Experimental Sauter diameters (full dots) are compared against the full 3D simulations with Ansys Fluent (solid lines). The evolution of the Sauter diameter for different viscosities (blue: 0.5 mPas, red: 12 mPas, green: 30mPas, magenta: 242 mPas) is reported against the physical (and simulation) time for 4000s. In black is reported the evolution of the Sauter diameter calculated with the Laakkonen kernel by varying the value of one the constants (C_3) from 0.2 to 0.15.*

A perfect match was found when the value of the constant was equal to 0.15, but when the value of the constant C_3 is varied, all the curves are affected. Small values of this constant (half of the original value), shift all the curve through smaller values of the diameter, because the resistance to the breakage is reduced because of the viscosity, meaning that the only parameter which is affecting this phenomenon is the difference in the interfacial tension between the phases (equals for the different oils). In this case, it is possible to see that the three curves at low viscosity almost collapse on a single curve, while the high viscosity curve underestimates value. A value of the constant of 0.15, perfectly matches the experimental data, even if also in this case, the low viscosity curves are underestimated. For higher values of the constant, the high viscosity curve is lost, while the low viscosities are easily matched. With these preliminary results, we tested our findings on MATLAB, running all the cases for different viscosities. Results

are reported in Figure 6.44.

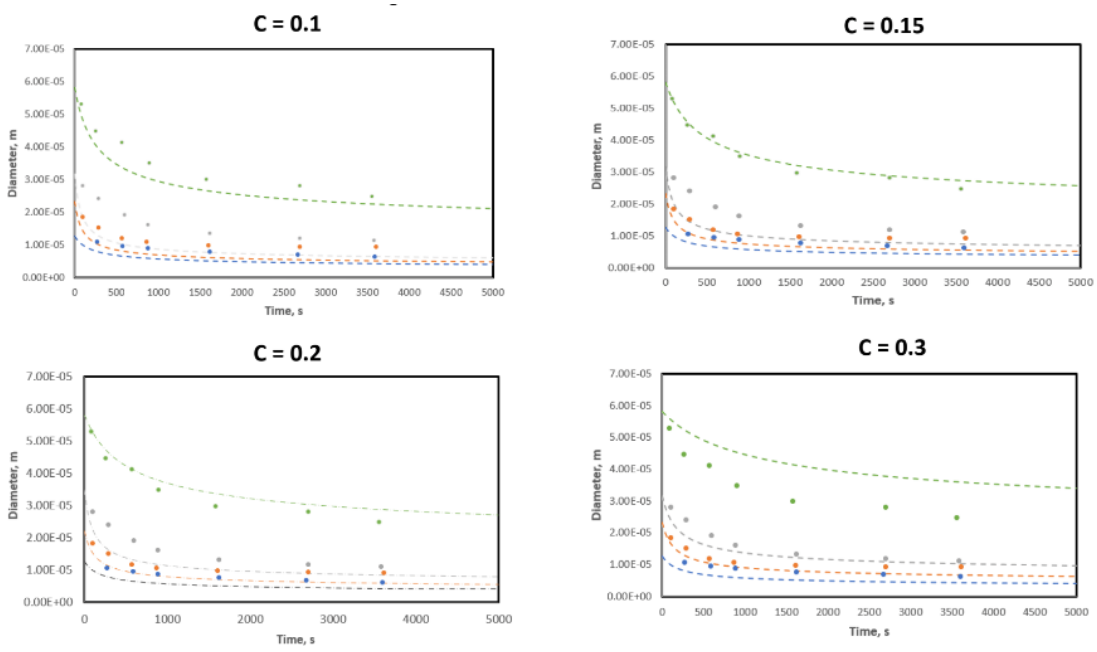


Figure 6.44. *Experimental Sauter diameters (full dots) are compared against the full 0D simulations with MATLAB (dashed lines). The evolution of the sauter diameter for different viscosities (blue: 0.5 mPas, orange: 12 mPas, gray: 30mPas, green: 242 mPas) is reported against the physical (and simulation) time for 4000s. In black is reported the evolution of the Sauter diameter calculated with the Laakkonen kernel by varying the value of one the constants (C_3) from 0.1 to 0.3.*

Three models can be used to face these problems. The first one is a full CFD-PBE approach, where CFD is solved together with the PBE at every timestep. The second and the third are an average and a lumped model. In the last two approaches, PBE is not solved for every cell at every moment, but the behaviour of a single cell representative of the whole system is looped through time. In the second method, the flow field is fully developed and since the disperse phase does not influence it (e.g. disperse phase has low viscosity), the field is frozen and only the PBE is solved in time. The first and second method gave us similar results. From these results, we can observe that the results are in better agreement with the simulations, using the Laakkonen kernel.The replacement of the epsilon value with its average can be used as an approximation but it is convenient to run full three-dimensional simulations. Discrepancies between low viscosity curves (using C_3 = 0.2) and the experimental data are below 5% or errors. In few points, we found an error of around 10%, but uncertainties are also present in the experimental data. The initial value of the populations that we used in the simulation was obtained assuming a log-normal distribution, with a variance of 15% of the initial diameter. Solid curves are representative of the equilibrium or Sauter diameter, obtained from

the ratio between the moments of order three and two of the droplet size distribution. For each physical timestep, equal to one second, we performed 300 internal iterations, ensuring that residuals were below 10^{-7} for all the moments. In particular we used as term of comparisons the value of the fifth moment, that is the slowest in convergence. Moments, that are representative of physical quantities related to the disperse phase, evolve during the simulation. As an example, we reported in Figs 6.45-6.46, moments of order zero and three in time. Moment of order zero represents the number of particles in the mixer, while the moment of order three represents the volume fraction. It is possible to appreciate that $M0$ increases over time because it represents the number of droplets that are generated due to the breakage phenomenon. Breakage is induced by the turbulence (high shear region) developed by the impeller:

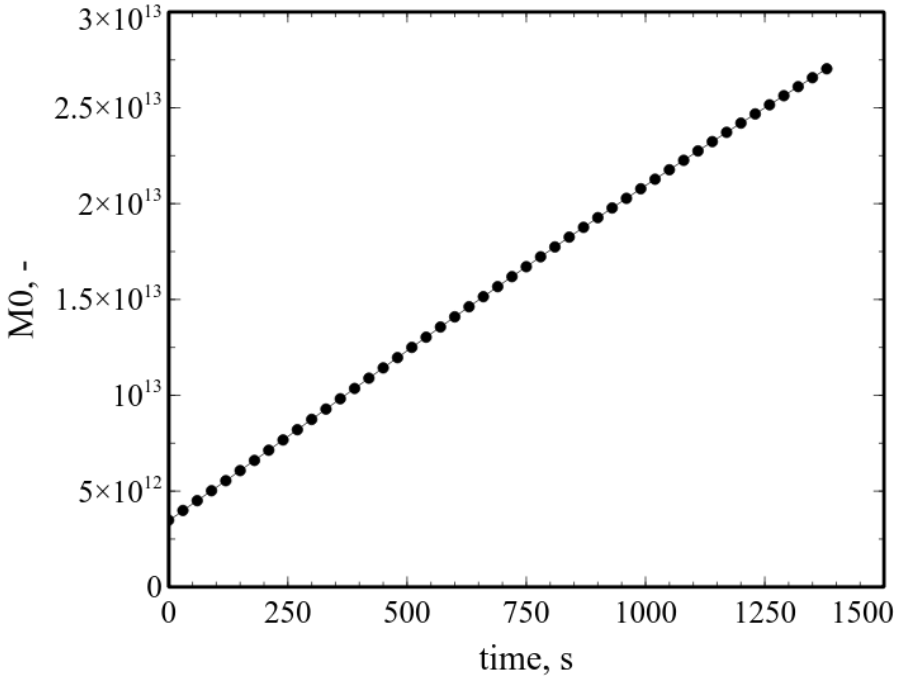


Figure 6.45. *Evolution of the moment of order 0, representing the number of droplets originated into the system, during the simulation time obtained with the Laakkonen kernel.*

The volume fraction, or M_3, of the disperse phase should not change during the whole simulation because no mass is added to the system. By observing the evolution of the moment three of the distribution it is possible to observe that no mass is lost during the resolution of the equations.

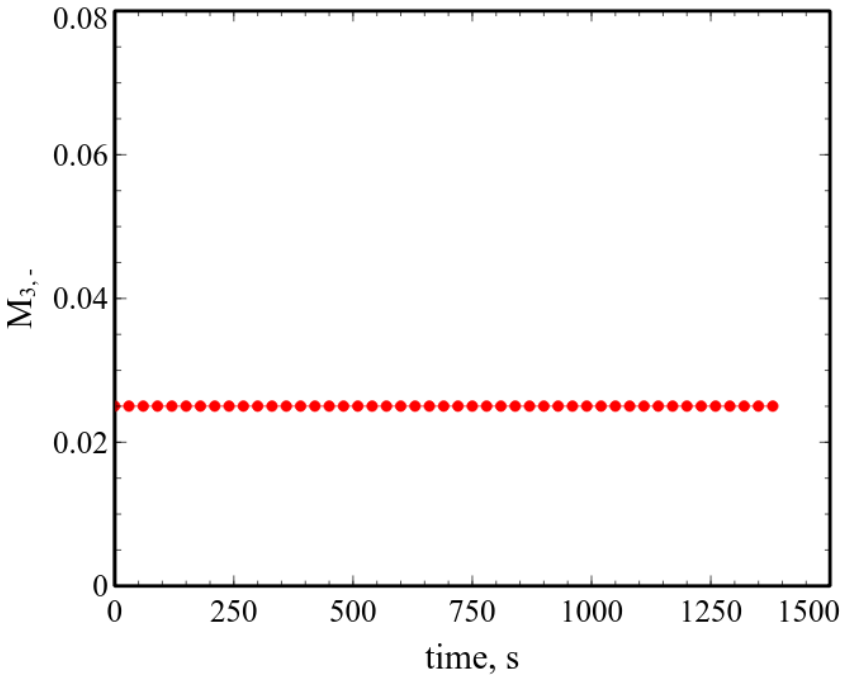


Figure 6.46. *Evolution of the moment of order 3, representing the volume fraction of disperse phase, during the simulation time obtained with the Laakkonen kernel. The conservation of moment of order 3 ensures that mass is always conserved along the simulation time.*

Coulaloglou and Tavlarides Kernel

We tested Coulaloglou and Tavlarides (CT) kernel, starting from the same initial conditions we used for the Laakkonen kernel, i.e. developed velocity field, uniform concentration of the disperse phase, and initial log-normal distribution of the population by assuming a variance of 15% of the initial diameter. Simulations run for at least 4000 seconds and moments were recorded every 30 seconds. The velocity field was decoupled from the PBE as it was done previously with the Laakkonen kernel. From our analysis, we could immediately see that CT kernel provides a higher breakage frequency, as demonstrated in Figure 6.47 and Figure 6.48. The number of particles in the system, increases much faster compared to Laakkonen kernel.

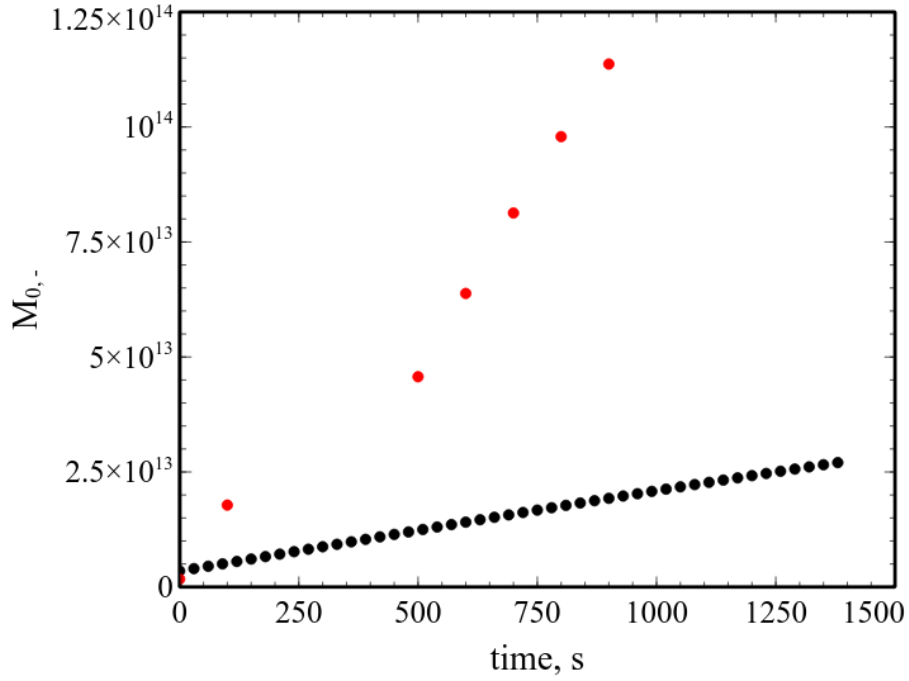


Figure 6.47. *Comparison between the moments of order 0 obtained with Laakkonen (black) and Tavlarides (red) kernel for a mid viscosity silicone oil in water. It is possible to appreciate how the two kernels produce a completely different dynamics.*

The breakage frequency is higher at the beginning and it decreases while the dimension of the particles decreased because of the breakage. The value of the frequency approaches zero when droplets are close to the Kolmogorov length. For such value, the frequency becomes too small, and the droplets do not break anymore. This asymptotic value can be already observed before 1000 seconds. The moment of order three is constant also for this case, meaning that the mass is conversed during the simulation.

For all the viscosities we performed simulations on Fluent and compared the results

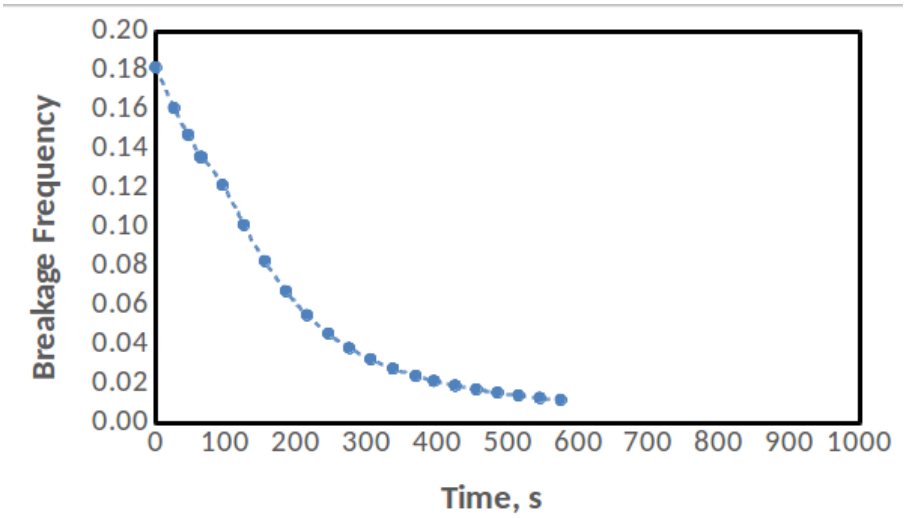


Figure 6.48. *Breakage frequency for the Tavlarides kernel plotted against the simulation time. It is possible to observe how the frequency rapidly goes to zero after few hundreds seconds of simulations, meaning that no breakup occurs after that limit.*

with the experimental data by El Hamouz. Now, we report the evolution of the Sauter diameter in time for different values of viscosity, Fig 6.49.

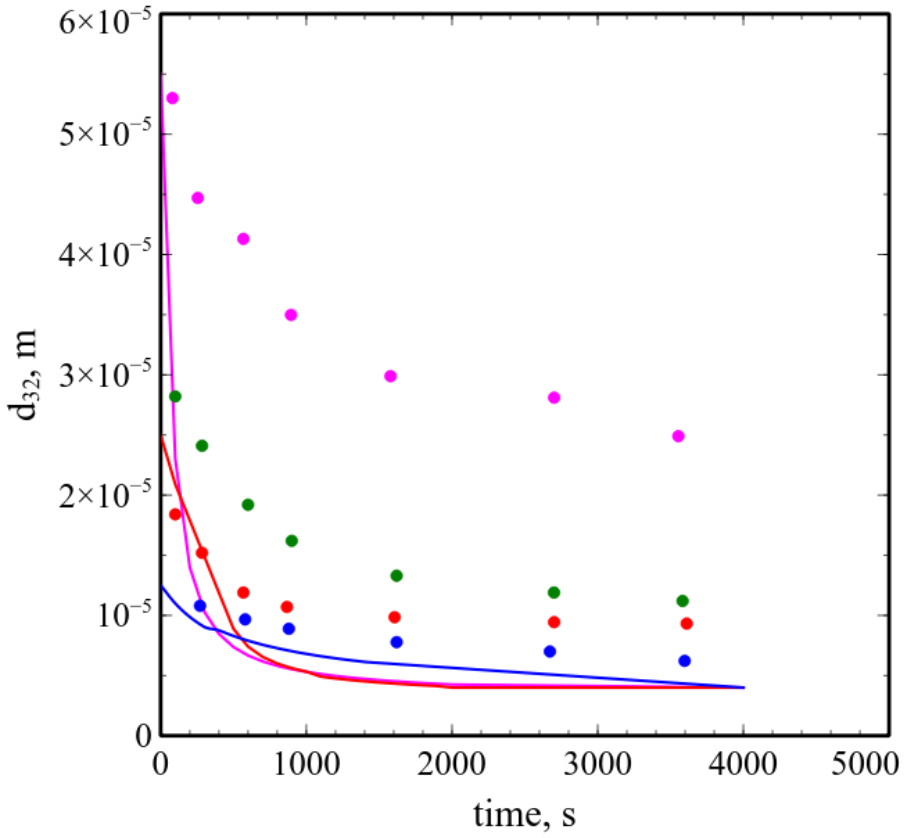


Figure 6.49. *Experimental Sauter diameters (full dots) are compared against the full 3D simulations with Ansys Fluent (solid lines) using Coulaloglou and Tavlarides kernel. The evolution of the Sauter diameter for different viscosities (blue: 0.5 mPas, red: 12 mPas, green: 30mPas, magenta: 242 mPas) is reported against the physical (and simulation) time for 4000s.*

The results obtained with CT kernel are now explained. Starting from different viscosities and diameters, all the curves collapse on one single line, in different times, because there is no contribution of the viscosity in the breakage frequency. This means that only the interfacial tension is playing an active role in the breakage phenomenon, but since all the silicone-oil/water mixtures have the same interfacial tension, no difference in the final diameters of the droplets can be reproduced. As it was demonstrated in the previous part, after 800 seconds, the breakage frequency goes almost to zero for all the cases and a stable mixture is obtained. More specifically, the dimension of the droplet is so small that we are almost at the Kolmogorov length (i.e. $5.5 \times 10^{-6} m$, close to the impeller region). We stopped these simulations after 3000 seconds because the breakage frequency was found to be negligible. We implemented and tested CT kernel in MATLAB and repeated the protocol we used for the Laakkonen kernel. In Fig 6.50, it is possible to observe that the simulation obtained in MATLAB

are exactly matching Fluent results for what concern the behavior of the kernel and the final value of the equilibrium droplet diameter. However, curves approach the zero value of the breakage way faster than Fluent simulations. This could be caused by an overestimation of the turbulent dissipation energy, introduced in the model in the form of average turbulent dissipation energy. The asymptotic value is reached after around 100 seconds for all the viscosities and all the curves collapse on a single one, representative of the Kolmogorov length scale. No tuning was performed at this stage, because of the different way in which kernels are formulated. By tuning the constants, it would be possible to capture the final value of the diameter while the dynamics could remain wrong for all the cases. Moreover, we did not want to tune constants for every specific situation but more finding fixed parameters to describe the mixing process as generally as possible.

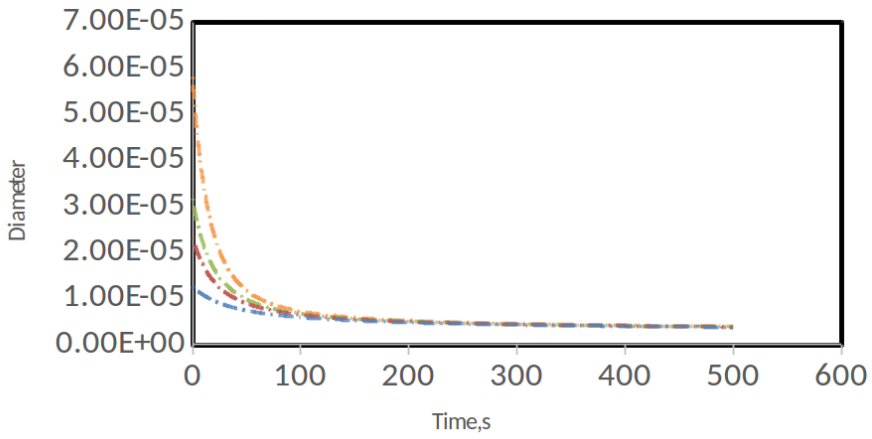


Figure 6.50. *0D simulations with MATLAB (dashed lines) using Coulaloglou and Tavlarides kernel. The evolution of the Sauter diameter for different viscosities (blue: 0.5 mPas, red: 12 mPas, green: 30mPas, orange: 242 mPas) is reported against the simulation time for 500s.*

A final comparison that highlight the difference between CT and LA kernel can be based on the different droplet distributions that can be reconstructed using the method proposed by Marchisio and Fox, 2013, where the log normal distribution parameters can be obtained by using the moments:

$$\mu = \frac{j}{ij - i^2} ln\left(\frac{m_1}{m_0}\right) + \frac{i}{ij - j^2} ln\left(\frac{m_j}{m_0}\right), \tag{6.7}$$

$$\sigma^2 = \frac{1}{1 - \frac{i}{j}} \left[\frac{2}{j^2} ln\left(\frac{m_j}{m_0}\right) - \frac{2}{ij} ln\left(\frac{m_i}{m_0}\right)\right], \tag{6.8}$$

where i=2 and j=3.

In particular, we selected the moments of order zero, two and three, since any of the moments can be used for this purpose. We compared the distributions for Laakkonen and CT kernels after 300 and 1200 seconds, for the low viscosity silicone-oil/water mixture, as reported in the following:

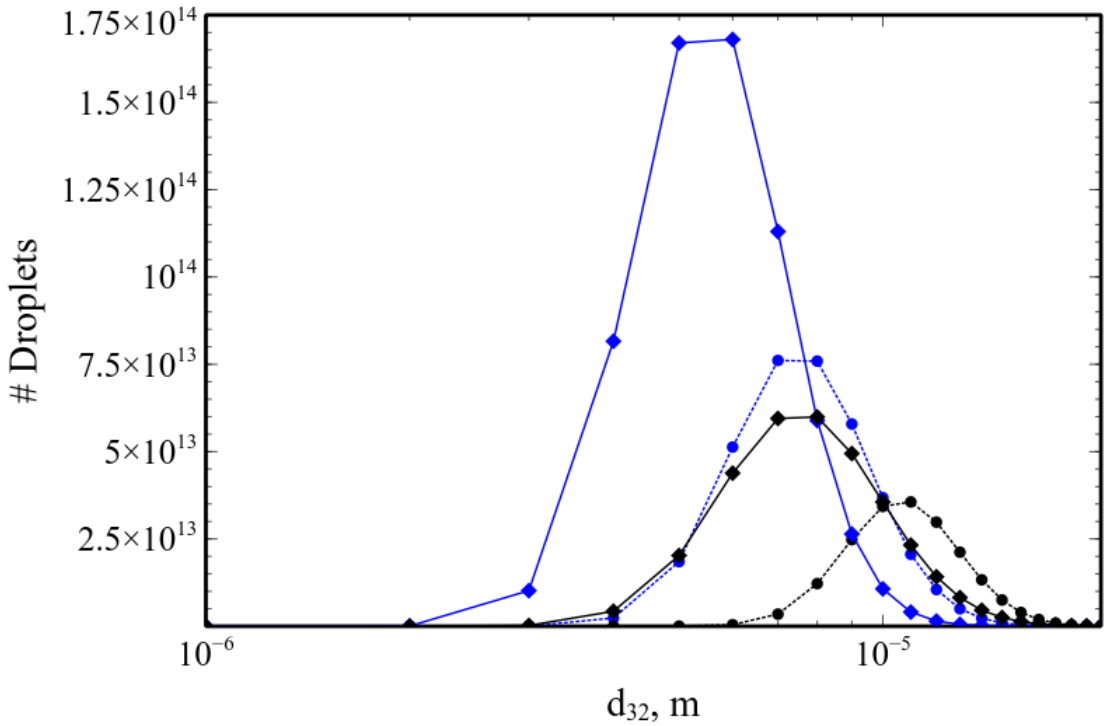


Figure 6.51. *Comparison between reconstructed droplet size distributions calculated from the moments, using CT (blue) and LA (black) kernels after 300s (dashed) and 1200s (solid) for a mid viscosity silicone oil in water.*

From these results it is possible to highlight how the effect of the viscosity term in the Laakkonen kernel is acting to reduce the breakage frequency, also slowing the overall process. CT kernel predicts the final value of the Sauter diameter after 800s as it was shown before, while the Laakkonen kernel reproduces slower breakage because of the presence of the viscosity term in the kernel. Since this term seems to be driving the overall process, we performed a sensitivity analysis on the two groups that are present in the Laakkonen kernel (and are omitted in CT). The aim of this analysis was to record the values of the two groups (reported below for clarity) for different values of viscosity, diameter and turbulent dissipation rate:

$$G_1 = C_4 \frac{\sigma_i}{\rho_c \epsilon_{turb}^{2/3} d^5/3}, \tag{6.9}$$

$$G_2 = C3 \frac{\mu_d}{\sqrt{\rho_c \rho_d} \epsilon_{turb}^{1/3} d^{4/3}}, \tag{6.10}$$

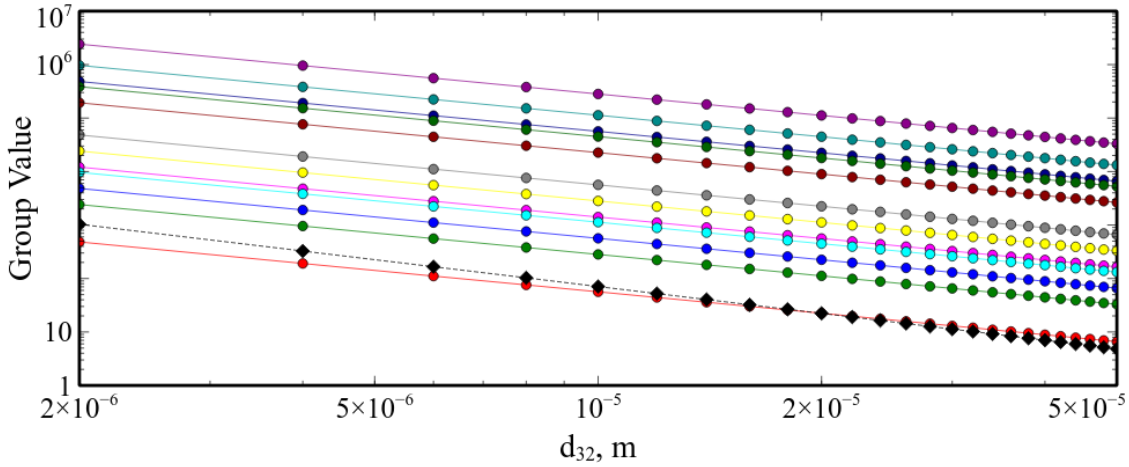


Figure 6.52. *Sensitivity analysis on the value of the two groups present in the Laakkonen kernel. The group containing the interfacial tension (dashed) is compared with the group containing the viscosity (solid) at different values of the viscosity (red: 0.1mPas, green: 0.5mPas, blue: 1mPas, light blue: 5mPas, pink: 10mPas, yellow: 20mPas, grey: 40mPas, dark red: 100mPas, dark green: 200mPas, dark blue: 400mPas, dark cyan: 500mPas, dark purple: 700mPas with* $\epsilon = 4.5 kgm^2 s^{-3}$

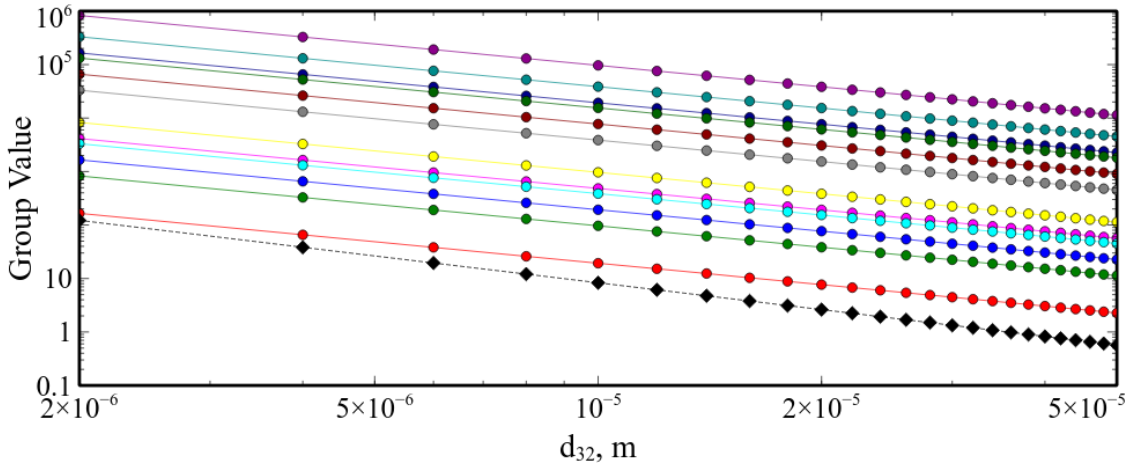


Figure 6.53. *Sensitivity analysis on the value of the two groups present in the Laakkonen kernel. The group containing the intefacial tension (dashed) is compared with the group containing the viscosity (solid) at different values of the viscosity (red: 0.1mPas, green: 0.5mPas, blue: 1mPas, light blue: 5mPas, pink: 10mPas, yellow: 20mPas, grey: 40mPas, dark red: 100mPas, dark green: 200mPas, dark blue: 400mPas, dark cyan: 500mPas, dark purple: 700mPas with* $\epsilon = 115 kgm^2 s^{-3}$

We can infer that the contribution of the group 1, containing the interfacial tension, is always less important than the term containing the viscosity. These values are part

of an error function complementary, meaning that higher values of the groups push the frequency to zero. Group 1 is only comparable in the case of low viscosity, while for all the others, this term becomes negligible in the range of diameters explored. We found that for the high viscosity case Group 2 is four orders of magnitude bigger than Group 1. This also helped us in reducing the number of constants that had to be tuned in our model, because no action was taken on C4 but also CT kernel, that has no term containing the viscosity of the disperse phase. It has to be remarked that the range of analysis where we performed our simulations is borderline with the viscous subrange. These kernels perform better when there is a clear segregation between scales. In particular, droplets should be smaller than the macroscale of the system, defined by the size of the biggest eddies that are generated from the impeller (usually macroscale length is in the order of magnitude of the diameter of the impeller), but bigger than the Kolmogorov length scale, that depends on the viscosity and turbulent dissipation rate. In our simulations we were extremely close to the Kolmogorov scale. If we cross this limit, phenomena involved in the breakage process are not captured anymore by CT and AL kernels, but more complex mechanism must be introduced.

6.3.2 Silverson Mixer

Silverson mixer is a rotor-stator mixer that is commonly used in industries to homogenize and further reduce the size of the droplets obtained by traditional stirring. It was extensively characterized from the experimental point of view by Kowalski et al (James, Cooke, A. Kowalski, et al., 2017; James, Cooke, Trinh, et al., 2017), such that accurate expressions for the power number were derived. These systems are extremely complex to simulate, especially in 3D cases, because of the high number of details that form the two screens. The power draw, i.e. the measured power, of a Silverson mixer reads as follow:

$$P = PO_z \rho N^3 D^5 + k_1 M N^2 D^2 + P_L \tag{6.11}$$

Where P, in Watt, is the power draw, PO_Z is the calculated power number with zero flow-rate, ρ is the density, N is the speed or rotation, D is the diameter of the impeller, k_1 is a constant that is obtained experimentally, M is the mass flow rate and P_L contains the loss of power due to the efficiency of the mixer. If one can exclude losses, it is possible to re-organize Eq 6.11 in a more engineering version, based on non-dimensional quantities, in terms of Power Number (PO):

$$PO = PO_z + \frac{k_1 M}{\rho N D^3} = PO_z + k_1 N_{Qv} \tag{6.12}$$

where N_{Qv} is the non-dimensional flow rate, or discharge number. In this way, the equation is a straight line, with slope equals to k_1 and intercept equals to PO_z, and these two values can be used to as a term of comparison between experiments and

simulations. We report in Fig 6.56 the velocity fields obtained for different velocities of the impeller and mass flow rate:

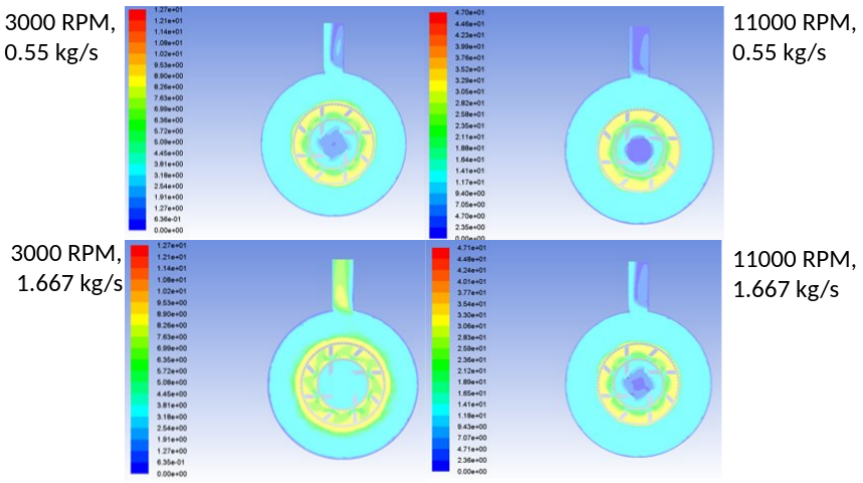


Figure 6.54. *Countour plots of the velocity fields obtained for the Silverson mixer at different velocities of the impeller and mass flow rate.*

In the specific case, it is possible to appreciate how the pattern of velocity field, in fully turbulent regime, does not change much in the different cases but in the case where the inlet velocity is at least comparable with the velocity of the impeller in the central part. Different meshes and turbulence models have been tested on the system and results are reported in Figs 6.55 and 6.56:

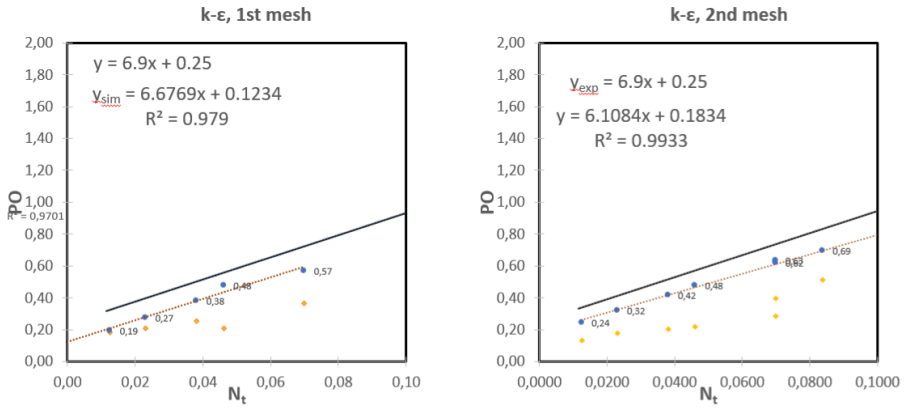


Figure 6.55. *Comparison between the power number obtained from the Torque and the turbulent dissipation rate, with $\kappa - \epsilon$ turbulence model for two different meshes (different level of refinement) compared against the experimental results obtained by Hall, Cooke, Pacek, et al., 2011; A. J. Kowalski et al., 2011. The black line is the experimental Power Number, while the red line represents the best simulation fitting curve. Slope and intercept of experimental line can be compared against the simulation ones. Blue symbols represent the PO obtained by torque, yellow dots represent the PO obtained by the turbulence.*

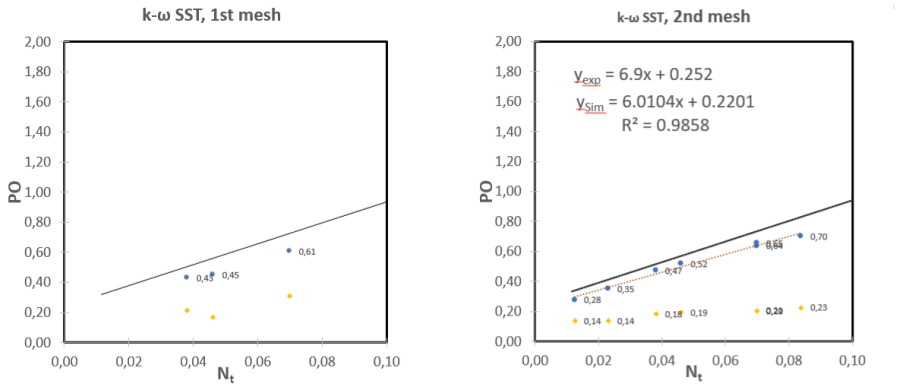


Figure 6.56. *Comparison between the power number obtained from the Torque and the turbulent dissipation rate, with $\kappa - \omega$ turbulence model for two different meshes (different level of refinement) compared against the experimental results obtained by Hall, Cooke, Pacek, et al., 2011; A. J. Kowalski et al., 2011. The black line is the experimental Power Number, while the red line represents the best simulation fitting curve. Slope and intercept of experimental line can be compared against the simulation ones. Blue symbols represent the PO obtained by torque, yellow dots represent the PO obtained by the turbulence.*

We compared the results obtained from different turbulence models, trying to reproduce the experimental curve of the pilot scale Silverson Mixer. The curves obtained in the four cases, with 2 different meshes with a number of elements which

goes up to 14 000 000 in the most refined case, reported an error between 10% and 15%, which can be considered reasonable if we assume that experimental curves for such systems may already have errors with them. The power number, obtained from the torque acting on the impeller in all the cases, matches with the experimental data even if the $\kappa - \omega$ model seems to provide better results due to the strong swirl effects that are reflected also in the velocity fields. It must be highlighted that when power number is obtained from the turbulent dissipation rate, the values are consistently wrong and they are extremely mesh dependent (this is not true in the case of the torque). All the curves obtained from the simulations lie on a straight line and this is in line with the experimental observations. We obtained our own equations for the different meshes and turbulence models (slope and intercept are reported in Table 6.6) and validated the power draw, the value of the dissipated power that can be read from the experimental setup, against the simulation results, as it is possible to appreciate in Fig 6.57:

Table 6.6: Slopes and intercepts obtained from the fitting of the experimental results of the Silverson 150/250 Mixer compared against the experimental curve

Case	PO_z	k_1	R^2
Experimental	6.9000	0.25	0.96(verif.)
Case 1	6.677	0.123	0.979
Case 2	6.108	0.183	0.993
Case 3	6.022	0.212	0.988
Case 4	6.010	0.220	0.989

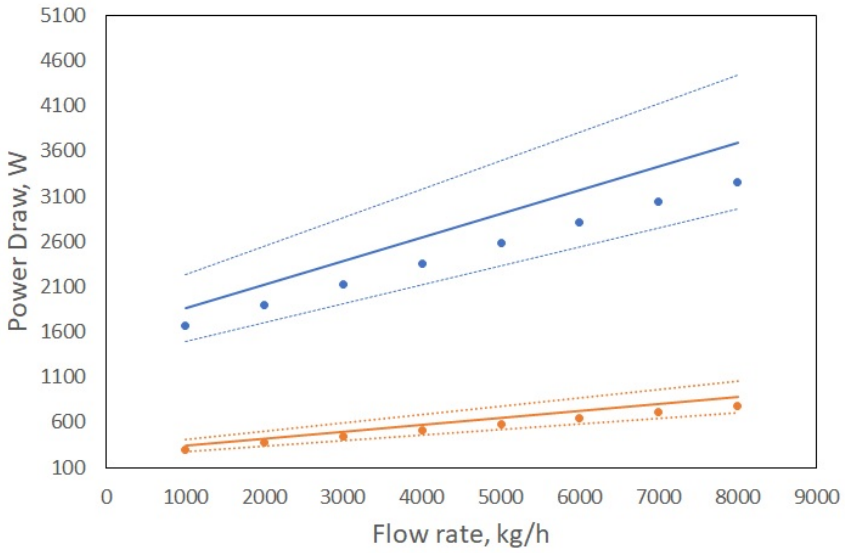


Figure 6.57. *Power draw obtained at different flowrates and speed of rotation of the impeller (blue: 11000 RPM, orange: 6000 RPM) obtained by experiments (solid line) and simulations (dots). Dashed lines represent an interval of confidence of 20% which is related to experimental uncertainties.*

And for the Laakkonen kernel, the Sauter diameter of the droplets, assuming a constant inlet of emulsion in the central part of the mixer can be appreciated in Fig. 6.58:

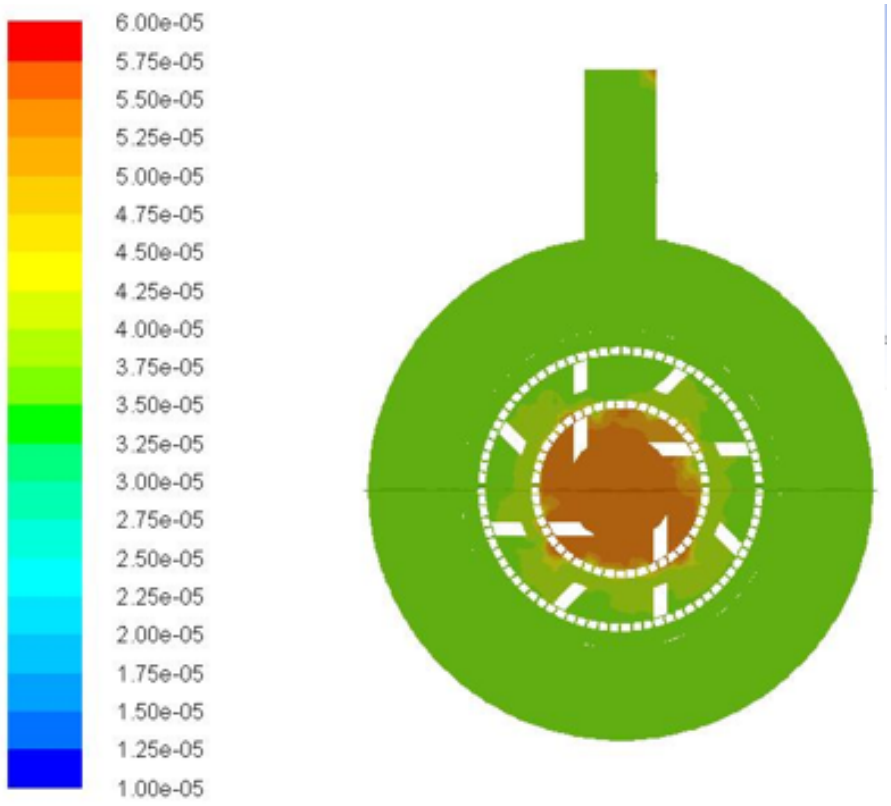


Figure 6.58. *Simulation results of the Laakkonen kernel for the silverson mixer. Sauter diameter is reported on a horizontal section plane of the mixer. Smallest reported diameter is in the order of magnitude of* $3 \times 10^{-5} m$.

These results are in agreement with the experimental setup that can be found in Hall, Cooke, El-Hamouz, et al., 2011 where a similar setup is proposed. The droplet diameter is assumed to be constant at the inlet region and decreasing by almost 50% in the outer regions.

6.4 Coupled Solver: CFD-DPD

In the final part of this work, a possible approach on the coupling between the two simulation scales is assessed. Before rushing into the description of the code, it is important to highlight why and how coupling is important and must be made. In many industrial cases, when one tries to describe the evolution of a system, microscopical phenomena might be avoided during the macroscopical simulation. However, those phenomena are fundamental in describing in the best possible way the outcome of a process, hence they must be addressed in our simulations. Our attempt tries to extract information from a microscopical level (i.e. mesoscale by using LAMMPS) and send

it to a macroscopical level (i.e. OpenFOAM), such that the evolution of the transport coefficient that is directly coupled to the microscopical state of the system can be appreciated and correctly taken into account rather than being approximated by empirical ad-hoc models. In this way, it is possible to overcome fine-tuned models and move to a complete description of a system. Attention must be paid to the increased number of calculations that have to be performed. If one runs a DPD simulation for each of the cells of the CFD domain, the computational cost will be huge (approx 4 hours per cell for our specific case). Moreover, the dimension of the cell must be small enough in order to be representative of the geometrical domain. One possible solution could be to simulate different conditions and store similar results in libraries, such that when one cell is identical to another, microscopical simulation could be avoided and results are simply recalled. In this way, an online (1 simulation for completely different scenarios) - offline (results are stored and can be recovered) tool can be used to improve multiscale simulations. With this idea in our mind, we developed an α-version of coupled DPD - CFD solver, starting from two open source codes, OpenFOAM and LAMMPS. Due to the differences between temporal and geometrical scales, the code works in a *decoupled* way, meaning that variables that are obtained at CFD scales are sent to the DPD scales while the CFD time is frozen, results are computed at the microscale and fed back to the CFD code, so that the macroscopical timescale can be updated. A schematic cycle of operation is given in the following:

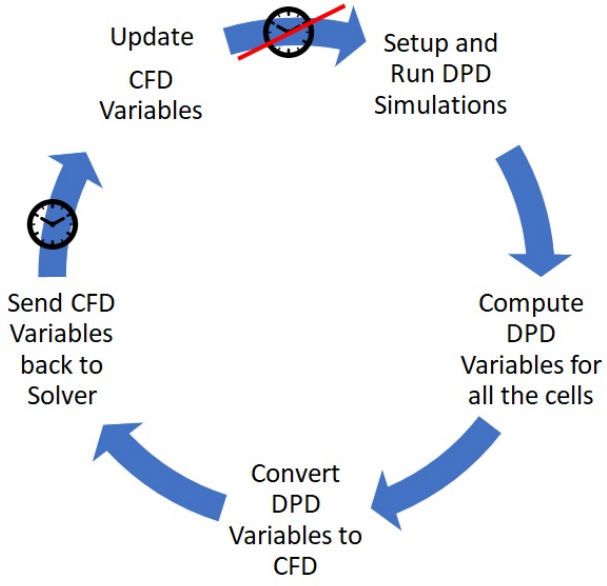


Figure 6.59. *Working flow of the coupled DPD/CFD code. In the first step, the DPD simulations are run and variables are computed for each cell of the CFD domain. DPD values are converted into physical units and loaded into the mesh. CFD time is updated together with the related fields. Time is stopped and DPD calculations are repeated.*

The choice of these two codes was related to the fact that both are written in C++, which made easier the practical linking, but any other code and language can be wrapped. OpenFOAM is a computational fluid dynamics code that has been widely used and validated against several cases in literature, while LAMMPS is a particle based code that contains a DPD package within it. We started from choosing the version 5.0 of OpenFOAM and a Non-Newtonian solver. In particular, we selected nonNewtonianicoFoam as a starting point, and for LAMMPS, we took the version of 16-Feb-2017, compiled with user DPD Package and as a dynamic library. To wrap the two codes into a single running one, we used a Message Passing Interface, and, in particular, MPICH was selected. The use of an MPI routine allowed us to run the code in parallel, using the same number of processors for both OpenFOAM and LAMMPS. LAMMPS was compiled as a dynamic library and called before any update of the velocity field within the OpenFOAM solver. The workflow can be observed:

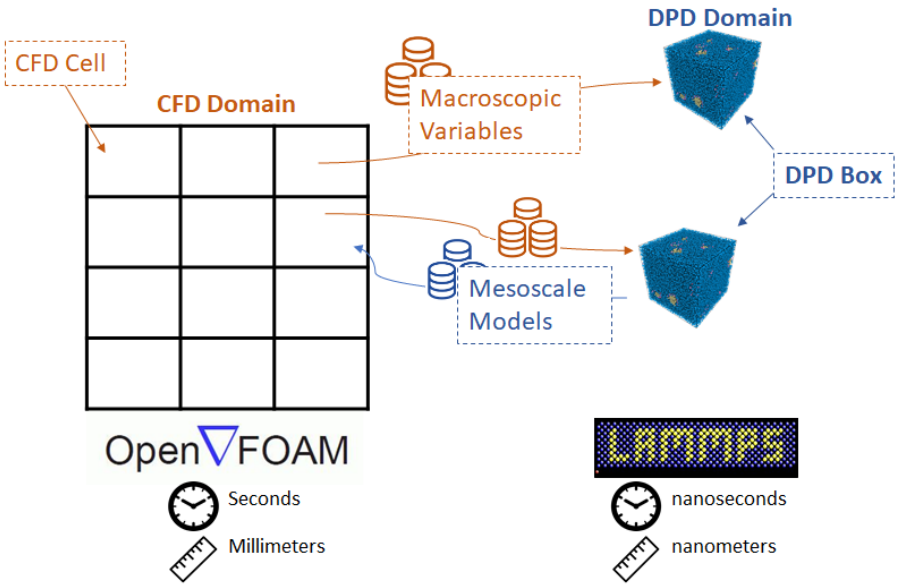


Figure 6.60. *Coupled solver overview. LAMMPS is linked to OpenFOAM. The cycle of operations previously illustrated is now given in a simpler way.*

MPI procedure must be initialized by defining the number of partitions on which OpenFOAM, and consecutively LAMMPS, have to be run. The CFD domain is divided into the desired number of bins, and the same number of procs will be used to run LAMMPS faster by re-organizing the use of the processors. An example of CFD-DPD simulation is now described in detail. A timer ensures that in the first N (a number defined by the user in a separate dictionary) timesteps, only the OpenFOAM routines are solved. The N indicator is increased after every timesteps and when it matches with the starting values, the DPD routine can be started.

We focused our attention on the calculation of the shear rate in each CFD cell. For

all the cells present in the domain, the shear value is rounded, according to a minimum level of precision defined by the user. All the rounded shear values are checked to avoid repeated values and are ordered and stored into a storage matrix, which is used as a library during the whole run. The number of independent shear is calculated and used as number of loops that must be performed by LAMMPS. For example, if in our cells we have 6 different values of the shear rate, we will perform 6 LAMMPS simulations. In this specific case, we ignore differences in concentration of the components and only shear is passed from the CFD domain to LAMMPS.

At this stage, the CFD time is stopped, and the LAMMPS simulations are run in series. The simulation time at this stage is driven by DPD simulations that must be performed until a pseudo-equilibrium (non-oscillating) value of the average viscosity is obtained. The code reads a standard input until the shear has to be indicated. At this stage, a string that is not contained into the LAMMPS setup file, is written by substituting the original DPD value with the OF shear, and passed to LAMMPS. The batch of simulations run, and at the end of every run, the viscosity of one box is extracted and stored, eventually the operation is repeated for all the values indicated into the initial loop. For all the shear values, the corresponding viscosities in DPD units are computed and stored into a library containing this value and the corresponding shear, so that if a matching shear is found in the future, no simulation will be run. The values of DPD shear must be converted into real physical units but at this stage we do not have a way to convert the units. If this step can be performed somehow, the CFD code can update the velocity field, by using the calculated viscosities for the different shear that have been identified into the system. When the velocity field is updated, the solver can proceed with the remaining OpenFOAM routines. In the next loop, if a new set of shear value is found, new LAMMPS simulations are run to obtain the new viscosities.

In this specific case we focused our attention on the shear rate, but any other variables can be passed between the codes and just some tweaks are needed to adapt the code. There is no direct connection between OpenFOAM and LAMMPS apart from the MPI routines that allowed us to share the processors for two different purposes. In fact, the LAMMPS routine can be extracted from nonNewtonianicoFoam and exported to any other solver. The alpha version of the code can be found at: https://github.com/hermessc/DPDCFD/tree/new_algo/src

6.5 Coupled Solver: 3D CFD - 0D CFD

In Fluent, it is possible to solve the PBE in different ways. Mathematical models are already available and ready to use, but they cannot be completely customized. However, in this specific problem, where it is important to have a clear understanding of the physics behind the phenomena that occur into the system, new kernels must be implemented in order to have reliable results. To run all the case, we used our

in-house implementation of the quadrature method of moments (QMOM), and we used 6 moments of the distribution, meaning that three nodes and weights were used to approximate the DSD. The Product Difference Algorithm was also implemented. The most important aspect in implementing our own solver for the PBE, is the possibility of including (or excluding) in a consistent way, the main phenomena involving the droplets, expressed as coalescence and breakage kernels. Three different models of breakage were coded and tested, and coalescence was set equal to zero, because of the presence of small amount of surfactant in the system (PPM of SLES).

Another important part of this work was related to the implementation of the same models into a MATLAB 0D code. According to Buffo (Buffo et al., 2016). it is possible to evaluate the evolution of the DSD with a much leaner implementation in MATLAB. All the previously discussed kernels and models were implemented into a zero 0D code, that simulates the evolution of DSD through time, once the distribution of the turbulent dissipation rate in the tank is given. For such model, it is important to underline that it can be used only in cases of small volume fractions, hence a strong separation between the droplet and the flow field scale is consistent (the velocity field is completely unrelated to the DSD, viscosity not affected by droplet diameter). The values of the turbulent dissipation rate were extracted from the fully turbulent developed flow field and a volume average was performed. Using this approximation, we reduce the simulation time and evaluated the final value of the droplets by varying the kernels, then also tested on Fluent. We also used this technique to perform tuning on some empirical constants that are present in the breakage kernels. The complete workflow is reported in Fig 6.61:

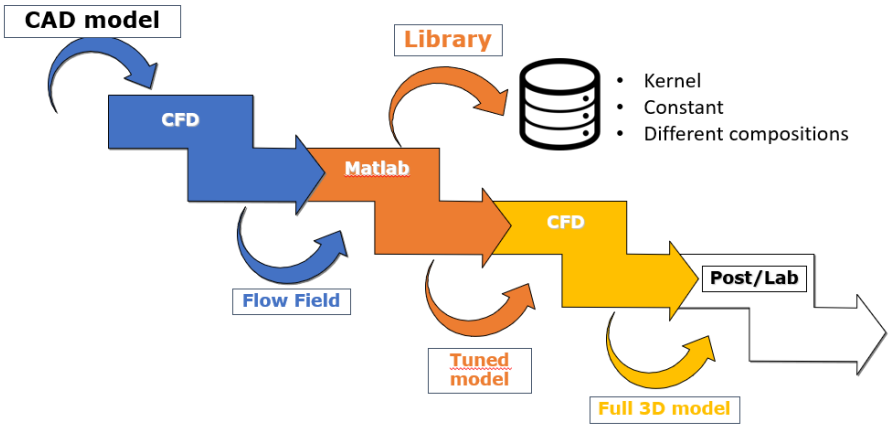


Figure 6.61. *Workflow of the procedure followed in the current work in order to reduce the number of full 3D simulations, without affecting the accuracy of the solution by coupling MATLAB 0D simulations and full 3D Fluent Simulations. In this case, the low fraction of the disperse phase allowed the simulation of a single cell by using the velocity field obtained from Fluent and test the different kernels to forecast the behavior of the DSD in time. This also allowed the tuning of the kernel constants without running full 3D simulations.*

In the proposed model to couple the codes, a CFD 3D simulation can be preliminary performed to develop the velocity field and extract the values of the turbulent dissipation energy. Local high shear values obtained around the impeller are the main contributors to the breakage phenomenon, which results dominant due to the composition of the mixture. The necessity of tuning variables that appear in the kernels are related to the high difference in the viscosities that were tested. A good performing kernel should include the dependence on the viscosity of the disperse phase, but the constant that are used have to be properly tuned in specific cases. Here comes the necessity of using the MATLAB code that speeds up the simulations and can be used as a benchmark tool to reduce the number of tests. Once the values of the constant are finely tuned through MATLAB, a full 3D model can be simulated and validated against the experimental data.

Chapter 7

Conclusions

This thesis is focused on the modelling and simulation of complex fluids on different scales. In particular, surfactants and emulsions have been investigated by using two approaches, mesoscale simulations (DPD) and macroscale simulations (CFD). These two approaches and the two systems can be switched between them (i.e. DPD can be used to simulate emulsions and CFD can be used to simulate surfactants). We also identified a path to link the two scales together in a single solver. In this final chapter, we want to summarize our findings but also highlight the limits we have discovered, the problems faced and the alternative solutions that may be used in future works.

7.1 Copolymers

Here we report our findings, strategies and limitations related to copolymers simulated with DPD:

On the simulation setup

Simulations of three different species were performed using the same set of parameters obtained from the literature. We were able to explore for three different systems (Pluronics L64, P84 and P104 in water), the complete spectrum of microphases, hence their phase diagrams. This means that DPD can represent a powerful tool that can be adopted in industries were different chemical formulations have to be tested. DPD simulations can be used to reduce the number of experiments, because it is possible to predict which microphase will be obtained by mixing different components. As demonstrated in this thesis, with DPD, by tuning only few specific parameters, it is possible to reproduce an incredible spectrum of mixtures.

However, the lengths of the investigated polymeric chains were different and this aspect has an important drawback in coarse-grained simulations. The tuned parameters could not work properly for extremely long polymeric chains since solubility parameters may differ from the selected one. In fact, repulsive coefficients

are calculated starting from the solubility of one single group into another single group, representing the molecules present into each cluster. However, when it comes to longer polymeric chains the behavior may differ from that of each single group. As an example, the Pluronics F127, which has a longer chain have to be properly tuned in order to reproduce its phase diagram.

Regarding the number of beads and chains necessary to reproduce one system, bigger boxes or higher DPD densities may also be required. A sufficient number of different structures is necessary in order to have statistical evidence and avoid artifacts due to the confined boundaries. A drawback of DPD simulations lies in the definition of the repulsive potential used between beads. Affinity between different species is only defined by tuning the repulsive coefficient of different species. This is a strong limitation that has to be coupled to the possibility of bead of interpenetrating and crossing bonds. Chains may cross each other because of the nature of the interactions. This is a strong limitation of the model that have to be overcome by the introduction of more complex models, that are able to describe cross-linking of polymeric chains.

On the microphases

If the attention is focused on the determination of the microphases, it was possible to define and characterize micelles both from a quantitative and qualitative point of view. This opens new possible scenarios, because the results given by this analysis can be translated into physical quantities, population balance equation can be extended to these problems and fully coupled with structural information obtained from DPD. A clustering algorithm must be used in order to assess a quantitative match with the experiments. We were able to define and verify the sphericity of the micelles in a certain area of the phase diagram, defined by the concentration of the disperse phase, by using information obtained from the gyration radius and the cluster mass distribution. Density based algorithm cannot be used at high concentrations of surfactant in water, because it recognizes all the structures as lumped in a unique blob. This aspect represents the main disadvantage of the algorithm but since the number of structures is not know a-priori, it was used to discern between many structures that may exhibit unclear shape. Only when a clear separation between the structures was present we were able to count the single microphases.

The use of such tool allowed us also to identify the hexagonal microphase. This structure was not obtained at equilibrium, but when shear was applied on the system, the disordered network aligns because of the streaming effect and reorientation becomes possible. The same concept applies to lamellae that could have been counted only when separation was neat. All these findings are fundamental in overcoming the actual macroscale modelling. In many cases, empirical evidences are still the main driver in obtaining tuned models. The possibility of exploring new scales, both in time and space, allows a better understanding of the relevant physics that rules those phenomena. In this way, it is possible to have a more general approach to

many different systems, without boundaries related to the experimental range of investigations. Also, very small scales, where relevant phenomena take place, are difficult to explore by using only experimental evidences and this is why simulation becomes a fundamental tool in predicting behavior of such complex fluids.

On the non-equilibrium limitations

We obtained viscosity curves that do not depend on the value of the spring constant and equilibrium distance. Potential was then switched to the FENE model. With such model, we ensured that any over-elongation was removed because of the formulation of this potential. The logarithmic term, present in the formulation of the potential, ensures that when particles move further than the accepted distance, the value of the recalling force goes to infinite. With these stiffer chains, we had to increase the simulation time, because of the relaxation time which was higher even for the equilibrium simulations. The viscosity curves however, behave similarly to those obtained using the Harmonic potential, even though small differences in the numerical values were found have to be investigated in detail. Artifacts introduced by shear effect were reduced by monitoring relevant variables.

The temperature of the system was kept bound to the original value and different thermostats have been tested in order to exclude any possible anomalous increment. A remark must be done at this point, in fact, not all the thermostats can be used in our problem. Berendsen thermostat, for example, implies that velocities of the beads into the system have to be rescaled in order to keep the temperature bounded to the thermostat values. If this is the case, the velocity profile, previously imposed on the box and developed by using LEBC, is lost because of the effect of the thermostat that rescales the velocities at every timestep. If a linear velocity profile is not achieved in the simulation box, it is not possible to extract information regarding the non-zero off-diagonal stress tensor component that we used to compute the viscosity of the DPD system.

On the shear effects

An important aspect that we were able to better understand was the fate of the microstructures when shear affects the system. Mixing is an important part in the manufacturing process of structured fluids. Mainly, all the different compounds are mixed together, by using different operating conditions, impeller and anchors, at different stirring rates and sufficient stirring time. Each small region in the mixing tank, experiences different shear rates due to the proximity of the impellers or the anchor. Complex flow fields can be translated into microscopic effects that can be assessed by using coarse-grained models. These models help identifying and predict how different microstructures react to different stirring intensity. Some structures or mesophases can be desired and obtained only under certain conditions. In our testcases, we were able to

assess the whole spectrum of concentrations and obtain relevant information regarding the modification of such microstructures. For example, spherical micellar structures do not modify their shape, keeping their original conformation and eventually coalesce if two of them come closer or break if they become to big. These effects, however do not change transport properties such as the viscosity of the system, which results constant. When higher concentrations of surfactant in water were investigated, simulations showed more complex phenomena. Disordered and complex microstructures that are present in equilibrium configurations, reorient themselves into more ordered patterns.

However, springs that connect different beads were tuned in order to avoid excessive elongation. In fact, beads that are contained in one single polymeric chain, are linked using specific form of potentials. One example can be provided by the harmonic potential, where only the strength and the cut-off distance for the interactions must be defined. In equilibrium simulations, these springs ensure that beads, that are part of a single chain do not move too far from each other, but when the system is affected by shear, since the value of these stresses can be extremely high, the spring imposed by the potential results too weak to keep beads close. If particles move away from their bounded neighbor, the identity of the chain is lost and the interactions between closer beads are lost if the neighbour moves further than the cut-off distance. Different potentials may be used, and in our work, we tuned the constant to reduce this over-elongation phenomenon. We tested a range of spring constants and cut-off distances and verified that these values do not alter equilibrium configurations. These new structures may exhibit a different rheological behavior compared to their initial microstate, and while shear is increased, properties, such as viscosity, may drop. This is a peculiar feature of Non-Newtonian fluids that have a viscosity that depends on the shear rate. Drops are not experienced for the system containing only water or low concentration of Pluronic L64 (up to 30%), while the Non-Newtonian behavior was observed at higher concentration. Experimental evidences of the system also confirmed these findings. The possibility of counting these structures gave also us the chance to find specific patterns. In particular we were able to identify the regions where micelles become elongated and their shape relax into oriented cylindrical type.

On the future work

No protocol has been identified to convert DPD units into real physical quantities in non-equilibrium. This means that at this stage, we were able to reproduce and understand from a qualitative point of view the transition between microphases, and the corresponding Non-Newtonian behaviour. However, the possibility of exploring such scales and retrieve these particular capabilities of structured fluids allow a possible interaction between data that is obtained at this level and models that are used to explore different time scales (e.g. the macroscale by computational fluid dynamics). It is clear that many possible parameters can be obtained and monitored regarding micelles in both equilibrium and non-equilibrium. Micelles can act as a population of

different objects that grow, coalesce, break and interact between them.

The capability of predicting and coupling these phenomena observed by DPD with real units, paves the way to new and more sophisticated models that are already present to describe populations from a macroscopic point of view. Empirical models could be replaced by more sophisticated models that take into account the real physics of underlying phenomena that are happening at microscale and that are truly considering both chemistry and physics of the system. A possible approach could be related to the use of non-dimensional numbers (e.g. based on the relaxation time of the microstructures), that should help the conversion between real and DPD units, at a more general and universal point of view. New DPD models have been released in these last years to overcome the problems previously illustrated, such as Smooth-DPD, by Espanol (Español and Revenga, 2003) that seems to be suitable for the direct comparison with real system. Computational models to describe the evolution of populations, without taking into account any information related to the microstructures were also included into this work and explored by CFD.

7.2 Emulsions

Here we report our findings, strategies and limitations related to emulsions simulated with CFD:

On the simulation setup

The aim of CFD simulations was to conclude and give a complete overview on the possibility of simulating complex fluids at different levels in order to obtain as much information as possible that can help and improve the design of mixing equipment. Stirring caused by impellers and anchors is the core aspect in manufacturing complex fluids that are used in food, personal care and home care industries. Thanks to CFD, it becomes possible to explore flow fields and specific patterns that are useful in identifying dead mixing zones. Mixing time is an extremely important parameter that tracks the rate at which a secondary phase is homogeneously mixed into the primary phase. By varying layouts (anchor, impellers, baffles...), mixing can be extremely different. Also, when a secondary phase is introduced into a continuous phase, the dimension of the droplets that are generated by mixing becomes a fundamental information because it drives many important phenomena, such as mass and heat transfer, viscosity, stability of the mixture. We simulated a system composed by water and an immiscible oil phase together with surfactant in order to derive the setup and verify those models used to describe the modification of the droplet size distribution. The system that we tested is a laboratory scale mixer, with a working capacity of 6 litres, and containing an anchor and an inclined sawtooth impeller. We tested small fractions of silicone oil with different viscosities and densities into water with the presence of parts per million of surfactant (SLES). In order to perform these simulations

using population balance approach, we started by reproducing the flow field of the mixer.

On the validation of the model

The given constructor power number was compared to different sets of simulations. In particular we tested many different approaches to validate the flow field developed by the rotation of the impeller. Different turbulence models have been validated on different meshes with increasing resolutions. MRF was used to reproduce the rotation of the fluid region surrounding the impeller. Another possible approach would have been the direct rotation of the mesh where there is contact between impeller and fluid regions. However, we kept a good refinement of the elements in the boundaries between rotating and fixed region, and imposed rotation of the fluid elements contained in a small region surrounding the impeller to reproduce this motion. The flow was slowly developed starting from lower rotation speed, up to fully turbulent region. This helped the convergence of the parameters, since starting at 3000 RPM caused crashes in the simulation. The convergence of the residuals continuity, velocity and turbulence parameters was kept below 1×10^{-6}, and sub relaxation factors have been used. Refining the mesh over a certain limit, by doubling the number of elements, was not worth the increased simulation time. It was observed that the power number obtained by the evaluation of the torque acting on the system and the power dissipated by the impeller was not changing if more elements were added. It was clear that fully hexagonal meshes gave more accurate results and reduced computational time, compared to the fully tetrahedral meshes. Ordered elements gave faster convergence and we decided to switch to fully hexahedral meshes for all the simulations. Results provided small differences between the computed power number and the constructor power numbers. Those differences can be attributed to a simple aspect that has to be considered. The calculation of the constructor power number is done by putting the impeller in vertical position in an empty tank. This is not the case of our mixer, where the impeller is inclined by eight degrees on the z-axis and anchor with baffles is present. However, all the simulations with different meshes and turbulence models gave us similar results regarding the computed power number, also varying the rotation speed of the impeller. We also tested the system by removing the anchor, and very small deviations from the original testcase were recorded.

When the disperse phase was introduced, we repeated the tests previously done on the single-phase system, by using the setup we identified to be the better performing. With the introduction of the disperse phase, turbulence was dumped by a small amount, but the general behaviour of the flow was not changed, and the power number was again found to be constant and similar to the previous cases. Disperse phase was patched into the system, assuming a uniform distribution of the oil, due to the previous effect of the stirring of the anchor. We re-computed the calculation of the flow field for all the different silicone oils (different viscosity and density). Again, residuals were taken

below 1×10^{-6} for all the relevant constants, and after that, PBE was solved. In our first attempt, we verified if the PBE could have been decoupled from the flow field. Due to the low volume fraction of the disperse phase and no effect of the droplet size on the viscosity of the mixture, the two testcases (fully coupled and decoupled) gave very similar results. We decided that the saving computational time (almost two minutes for each timestep. Simulation time of one timestep fully decoupled was three minutes and thirty seconds) was a priority due to the number of timesteps that had been simulated. Also, no advantage was found in running a fully coupled solver. Even if the coupling was not necessary, we updated every 50 simulated seconds the velocity field by assuming the correct DSD in fully decoupled solver. It has to be remarked that, when the volume fraction of the disperse phase is higher (e.g. vol. frac. > 1%, usually up to 10%), solving PBE and velocity field in a decoupled way is not possible, because the size of the droplets may affect the viscosity of the mixture, hence the velocity field developed by the impeller can have a different pattern and a drawback on the evolution of the DSD.

On the model tuning

By assuming the decoupling between scales, we were also able to run simulations by using the MATLAB code, as an example of the evolution of the DSD in one single cell. Each MATLAB simulation was almost hundred times faster than the full CFD simulation. This gave us the possibility to implement, explore and tune, if needed, different kernels to describe the breakage. In all the cases, we removed any coalescence effect, because the presence of SLES in the system, which is a very common surfactant used in many products, inhibit this phenomenon. In our simulations, we implemented and tested mainly three different kernels, in order to predict the evolution of the droplet size distribution. The three kernels, Laakkonen-Alopaeus, Coulaloglou-Tavlarides and Podgorska-Baldyga, are able to predict relevant phenomena only in the inertial subrange. If the droplets are too small (i.e. their dimension is below the Kolmogorov length scale), new models, that are able to describe the viscous sub-regime, have to be implemented and tested. In our testcases, experiments reported equilibrium droplet size that was always above the Kolmogorov length scale, hence we were able to use the previously discussed kernels.

On the model limitations

A comment on the use of the Laakkonen kernel must be done. In this kernel, the term preceding the error function, contains the turbulence dissipation energy and a constant. If we perform a dimensional analysis and compare Laakkonen and Tavlarides, we can see that in CT kernel, there is a diameter at denominator, such that the pre-exponential term has the dimension of a frequency. This does not apply to LA, where instead the empirical tuning constant is dimensional. This element has raised many

arguments among the experts of the area, and here comes the necessity of finding more universal forms of the kernels. LA kernel, even though gives better results, lack of some physics. Its final equation was indeed obtained by modelling many different empirical tests. This does not mean that the kernel is wrong, and actually the predictions of LA are able to detect phenomena that are ignored by the other kernels, that are obtained only thanks to these empirical secondary effects. This comment can be also used to explain why the kernel does not predict accurate results when the viscosity of the disperse phase is higher that a certain value. The system and the solution of the PBE model that has been used, ensures that when a particle breaks, two identical daughters are generated (similar volume and shape, mass is conserved). Also, we always assumed a log-normal distribution for the droplets that are obtained in our system, even at high viscosity. The kernel and the PBE is then able to reproduce the evolution of the DSD, for structured fluids if these hypotheses are valid. However, this is not the case of structured fluids with high viscosities.

First of all, we should carefully analyze how droplets break in the case of high viscosity fluids. Experimental evidences prove that the breakage is not binary, but instead from each droplet, due to extreme elongation and deformation resisted by the high viscosity, an high number of droplets can be formed. In general two or three bigger droplets may be obtained and surrounded by hundreds of smaller ones.

This behavior is however lost when the concentration of the disperse phase becomes higher, and binary breakage can be assumed if the concentration of the disperse phase is high (50% or more). The differences in the breakage event, may also cause the DSD to not follow the log-normal distribution but to become bimodal or even more complex. In cases of extremely high viscosity the distribution becomes extremely wide, and it spans over orders of magnitude. For these reasons the way in which PBE has been used in this work, may be able to capture only one peak of these anomalous distributions, which is, indeed, quite far from the d_{32} diameter predicted in the experiments. So if one wants to use the kernels to describe such complicated systems, fine tuning are required on the empirical constants, in order to reproduce and include those extra phenomena that are not explicitly accounted for in the original form of the kernel. This is not the case of the low-viscosity oil, where the distributions may be described as log-normal and the breakage event can be assumed binary without committing any particular error. For those cases, LA kernel that has been extensively tested on similar mixtures but on different testcases, is able to perfectly reproduce the evolution of the DSD in time for all the mixtures. In future works, the use of the BP kernel could overcome the limitations related to the empirical nature of LA kernel.

Appendix A

Codes

A.1 LAMMPS file

In this appendix, a simulation setup file is reported. Tests can be performed with LAMMPS 17_Feb_2016.

pluronic.lmp

Listing A.1: pluronic.lmp

```
1  units  lj
2  variable        ndim   equal  3
3
4  variable        xsize   equal  30
5  variable        ysize   equal  30
6  variable        zsize   equal  30
7
8  variable rhop  equal 3
9  variable plurchain equal   15
10 variable npart  equal (${xsize}*${ysize}*${zsize})*${rhop}
11 variable percn  equal 0.55
12 variable  wBea   equal (1-${percn})*${npart}
13 variable        plBeads equal   ${percn}*${npart}/${plurchain}
14
15 variable  kb  equal 1.0
16 variable  T  equal  1.0
17 variable        cutoff  equal  1.0
18 variable        gamma   equal  4.5
19
20 timestep 0.01
21 dimension       3
22
23 atom_style      hybrid bond dpd
24 boundary        p p p
```

```
25 comm_modify  vel yes cutoff 1
26
27 lattice  none 1
28 region  mybox prism 0 ${xsize}  0 ${ysize} 0 ${zsize} 0 0 0
29 create_box       3 mybox bond/types 3 extra/bond/per/atom 1
30 molecule polymer molecule.txt
31
32  create_atoms 1 random ${wBeads}  123445 NULL
33 create_atoms 1 random ${plBeads} 14567  NULL  mol polymer 100766
34
35 group water   type 1
36 group polymer type 3
37
38 mass * 1.0
39
40 neighbor  1.0  bin
41 neigh_modify  delay 0 every 6 check no
42
43 bond_style       harmonic
44 bond_coeff   1    40.0     1.0
45 bond_coeff 2  40.0    1.0
46 bond_coeff   3                    40.0    1.0
47 pair_style       dpd ${T} ${cutoff} 928948
48
49 pair_coeff    1 1  25.0    ${gamma}
50 pair_coeff  1 2  26.05  ${gamma}
51 pair_coeff  2 2  25.0  ${gamma}
52 pair_coeff  1 3  38.4  ${gamma}
53 pair_coeff   2 3   48.9  ${gamma}
54 pair_coeff  3 3 25.0 ${gamma}
55
56 thermo  1000
57 minimize  1.0e-5 1.0e-7 1000 10000
58
59 variable  srate equal 0.01
60 variable velramp equal ${srate}*${ysize}
61
62 dump   first  polymer xyz 5000 video.xyz
63
64 fix   1 all nve
65 run       200000
66 velocity all ramp vx 0 ${velramp} y 0 ${ysize}
67 fix  shear all deform 1 xy erate ${srate} remap v flip yes units box
68 variable visc equal -pxy/(v_srate)
69 fix  vave all ave/time 10 100 1000 v_visc ave running start 000
70 dump   dumpnew polymer xyz 5000 video2.xyz
71 thermo_style  custom step temp press pxy v_visc f_vave
72
73 run  1000000
```

molecule.txt

Listing A.2: molecule.txt

```
1
2  # Triblock Copolymer: Pluronic 64 ABA 3-9-3
3  15 atoms
4  14 bonds
5  0 angles
6  0 dihedrals
7  0 impropers
8
9  Coords
10 1  0  0  0
11 2  1.0  0  0
12 3  2.0  0  0
13 4  3.0  0  0
14 5  4.0  0  0
15 6  5.0  0  0
16 7  6.0  0  0
17 8  7.0  0  0
18 9  8.0  0  0
19 10  9.0  0  0
20 11  10.0  0  0
21 12  11.0  0  0
22 13  12.0  0  0
23 14  13.0  0  0
24 15  14.0  0  0
25
26 Types
27
28 1  1
29 2  1
30 3  1
31 4  2
32 5  2
33 6  2
34 7  2
35 8  2
36 9  2
37 10  2
38 11  2
39 12  2
40 13  1
41 14  1
42 15  1
43
44 Bonds
45
46 1 1 1 2
```

```
 2  1  2  3
 3  2  3  4
 4  3  4  5
 5  3  5  6
 6  3  6  7
 7  3  7  8
 8  3  8  9
 9  3  9  10
 10  3  10  11
 11  3  11  12
 12  2  12  13
 13  1  13  14
 14  1  14  15
```

A.2 Python Clustering Algorithm

fnc.py

```
#Class:  TimeStep
#Comment: It contains positions for different PPO atoms at different
    timesteps)
#Param:
#  @number: the "label of the atoms"
#  @xCoord: x position of one atom
#  @yCoord: y position of one atom
#  @zCoord: z position of one atom
#  @beadHisto: number of beads per cluster
#  @chainHisto: number of chains per cluster
class TimeStep(object):
 number = []
 xCoord = []
 yCoord = []
 zCoord = []
 beadHisto = []
 chainHisto = []

#Function:
#Comment: Initialize the object. It assigns the lists to the object
 def __init__(self,number,xCoord,yCoord,zCoord,beadHisto,chainHisto):
  self.number = number
  self.xCoord = xCoord
  self.yCoord = yCoord
  self.zCoord = zCoord
  self.beadHisto = beadHisto
  self.chainHisto = chainHisto


#Function:
#Comment: #Function to assign the values read from the file to the
    objects
```

```
31 #Param:
32 # @self: reference object
33 # @atoms: label of the atom
34 # @xCoord: x coordinate
35 # @yCoord: y coordinate
36 # @zCoord: z coordinate
37  def add_atom(self,atoms,mxCoord,myCoord,mzCoord):
38    self.number.append(atoms)
39    self.xCoord.append(float(mxCoord))
40    self.yCoord.append(float(myCoord))
41    self.zCoord.append(float(mzCoord))
42
43 #Function:
44 #Comment: It takes a list and the number of clusters and create a
       small matrix which counts all the occurrencies of a value into the
       list
45 #Param:
46 # @histo: list containing all the cluster sizes
47 # @size: number of clusters
48 #Return:
49  #@matrix: the matrix counting the occurrencies
50 def AverageManager(histo,size,matrix):
51  columns = 2
52  for cluster_size in histo:
53   for mini_row in range(0,5000):
54    if (matrix[mini_row][0] != cluster_size) and (matrix[mini_row][0]
         == 0) and (matrix[mini_row][1] != 1):
55     matrix[mini_row][0] = cluster_size
56     matrix[mini_row][1] += 1
57     break
58    elif (matrix[mini_row][0] == cluster_size):
59     matrix[mini_row][1] += 1;
60     break
61  return (matrix)
62
63
64 def sortingHistograms(normalized_histogram_x, normalized_histogram_y):
65  ordered_normalized_histogram_y = []
66  ordered_normalized_histogram_x = sorted(normalized_histogram_x)
67  for x in range(0,len(ordered_normalized_histogram_x),1):
68    for y in range(0, len(normalized_histogram_x),1):
69     if (normalized_histogram_x[y] == ordered_normalized_histogram_x[x
         ]):
70      ordered_normalized_histogram_y.append(normalized_histogram_y[y])
71  return ordered_normalized_histogram_y
72
73 def reBuildingHistograms(ordered_normalized_histogram_x,
     ordered_normalized_histogram_y):
74  reBuiltHisto = []
75  for x_element in range(0,len(ordered_normalized_histogram_x),1):
```

```
76    bin_size = int(round(ordered_normalized_histogram_y[x_element] * 10)
          )
77    for y_element in range(0, bin_size, 1):
78     reBuiltHisto.append(ordered_normalized_histogram_x[x_element])
79   return reBuiltHisto
80
81
82  def pbc_distance_matrix(coordinates,box):
83   import numpy as np
84   xdistance = 0.0
85   ydistance = 0.0
86   zdistance = 0.0
87   distance = 0.0
88   matrix_dimension = len(coordinates)-1
89   sq_dist = 0.0
90   j=0
91   distance_matrix = [[0 for x in range(matrix_dimension)]for y in range
         (matrix_dimension)]
92   for i in range (0,matrix_dimension,1):
93    j=0
94    if i != j:
95     while j < i:
96      xdistance = coordinates[i][0] - coordinates[j][0]
97      if xdistance > 0.5*box:
98       xdistance -=box
99      ydistance = coordinates[i][1] - coordinates[j][1]
100     if xdistance > 0.5*box:
101      ydistance -=box
102     zdistance = coordinates[i][2] - coordinates[j][2]
103     if xdistance > 0.5*box:
104      zdistance -=box
105     sq_dist = xdistance**2 + ydistance**2 + zdistance**2
106     distance = np.sqrt(sq_dist)
107     distance_matrix[i][j] = distance
108     distance_matrix[j][i] = distance
109     j+=1
110  return distance_matrix
111 def average_histograms_in_time(times, starting_interval,
       ending_interval, frameskip, frame_collection):
112  AverageMatrix  = [[0 for x in range(2)]for y in range (5000)]
113  #print ("inside and outside")
114  import matplotlib.pyplot as plt
115  import matplotlib.mlab as mlab
116  clusterRowOne    = []
117  clusterRowTwo    = []
118  average_histogram  = []
119  average_histogram_x  = []
120  average_histogram_y  = []
121  compatibility_histogram = []
122  ending_interval  = int(ending_interval/frame_collection)
```

```
123  corrected_interval  = int (starting_interval/frame_collection)
124  print (ending_interval)
125  print (corrected_interval)
126  print ((ending_interval-(corrected_interval)))
127  for counter in range(corrected_interval,ending_interval+1,1):
128   histogram  = times[counter].chainHisto
129   size   = len(histogram)
130   tempMatrix  = AverageManager(histogram,size,AverageMatrix)
131   #print (counter)
132   for x in range (0,size,1):
133    if(AverageMatrix[x][0] != 0):
134     clusterRowOne.append(AverageMatrix[x][0])
135     clusterRowTwo.append(AverageMatrix[x][1])
136   for x in range(0,len(clusterRowTwo),1):
137    average_histogram_y.append(clusterRowOne[x]*clusterRowTwo[x])
138    average_histogram_x.append(clusterRowOne[x])
139   for x in range(0,len(clusterRowTwo),1):
140    for y in range(0,int(clusterRowTwo[x]),1):
141     average_histogram.append(clusterRowOne[x])
142  #print (AverageMatrix)
143  print("End")
144  fig_3     = plt.figure()
145  dx     = fig_3.add_subplot(111)
146  min_hist   = 0
147  max_hist   = 0
148  hist_bin   = 0
149  min_hist   = min(average_histogram)
150  max_hist   = max(average_histogram)
151  hist_bin   = max(10,int(max_hist - min_hist))
152  from scipy.stats import norm
153  import matplotlib
154  (mu,sigma)   = norm.fit(average_histogram)
155  n,bins,patches   = dx.hist(average_histogram,normed=1,facecolor='
         green',alpha=0.75,bins='auto')
156  y    = mlab.normpdf( bins, mu, sigma)
157  l    = dx.plot(bins,y,'r--',linewidth=2)
158  plt.title('Histogram: cluster size')
159  plt.xlabel('Dimension')
160  dx.set_xlim(xmin=0)
161  plt.ylabel('Frequency')
162  plt.grid(True)
163  plt.savefig('average_dist_'+str(ending_interval*frame_collection)+'.
         png')
164  plt.close()
165  fig_4    = plt.figure()
166  ex    = fig_4.add_subplot(111)
167  ordered_normalized_histogram_x = []
168  ordered_normalized_histogram_y = []
169  max_y    = max(average_histogram_y)
170  normalized_histogram_x  = average_histogram_x
```

```
171 normalized_histogram_y  = [ y / max_y for y in average_histogram_y]
172 ordered_normalized_histogram_x = sorted(normalized_histogram_x)
173 ordered_normalized_histogram_y = sortingHistograms(
        normalized_histogram_x,normalized_histogram_y)
174 compatibility_histogram = reBuildingHistograms(
        ordered_normalized_histogram_x, ordered_normalized_histogram_y)
175 ex.plot(ordered_normalized_histogram_x,ordered_normalized_histogram_y
        , color='r', marker = 'o', alpha = 0.75)
176 plt.title('Normalized Plot')
177 plt.xlabel('Size')
178 plt.ylabel('Frequency')
179 plt.grid(True)
180 plt.savefig('normalized_plot_'+str(ending_interval*frame_collection)+
        '.png')
181 plt.close()
182 fig_5 = plt.figure()
183 fx = fig_5.add_subplot(111)
184 min_comp_hist   = 0
185 min_comp_hist   = min(compatibility_histogram)
186 max_comp_hist   = max(compatibility_histogram)
187 bin_comp_hist   = max(10,int(max_comp_hist - min_comp_hist))
188 fx.hist(compatibility_histogram,bins='auto',facecolor='green',alpha
        =0.75)
189 plt.title('Normalized Hisrogram')
190 plt.xlabel('Mimmo')
191 plt.ylabel('Alessio Domenico Lavino')
192 plt.grid(True)
193 plt.savefig('normalized_dist_'+str(ending_interval*frame_collection)+
        '.png')
194 plt.close()
```

cluster.py

```
#!/usr/bin/env python3
#!/usr/bin/python3
# Evaluation of number of clusters for different timesteps in a xyz
    file.

from fnc import *
#from gui import *
ppo = 9
box = 30
conc = 0.05
frameskip = 2
flag_hist = 0
flag_ave = 0
flag_pbc = 0
interval = 250000
RHO     = 3
FRAME_COLLECTION  = 500
PLURCHAIN  = 15
timecounter    = []
CheckCluster    = []
check_status    = 0

import sklearn
import numpy as np
from sklearn import cluster
from sklearn.cluster import DBSCAN
from sklearn import metrics
from sklearn.datasets.samples_generator import make_blobs
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings("ignore")
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import errno
import subprocess
import os
import math as mth
from math import sqrt
import traceback
import sys
import pylab as pl
from scipy.spatial.distance import pdist, squareform
import scipy
from collections import Counter
sys.tracebacklimit = None
filename = "video.xyz"
times    = [TimeStep([],[],[],[],[],[]) for _ in range(5000)]
print ("I am collecting coordinates")
```

```
xBeads  = box*box*box*RHO*conc/PLURCHAIN
Beads  = xBeads * ppo
coordinates = []

xyz = open(filename,"r")
coordinate_counter = 0
for line in xyz:
        try:
            atom, x, y, z  = line.split()
            index   = int (coordinate_counter / int(Beads))
            times[index].add_atom(atom, x, y, z)
            coordinate_counter += 1
        except ValueError:
            pass
xyz.close()
box_norm = 0
clusterSize = []
clusterGyrRad = []
print ("I am creating clusters and plotting ...")
for j in range (0,index+1,frameskip):

 coordinates   = np.column_stack((times[j].xCoord,times[j].yCoord,
     times[j].zCoord))
 coordinates_norm  = coordinates/box
 distance_matrix = [[0 for x in range(len(coordinates[0]))]for y in
     range (len(coordinates[0]))]
 if flag_pbc == 1:
  box_norm = box / box
  distance_matrix = pbc_distance_matrix(coordinates, box)
  db = DBSCAN(eps =2.2, metric='precomputed').fit(distance_matrix)
  core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
  core_samples_mask[db.core_sample_indices_] = True
  labels = db.labels_

 else:
  box_norm = box / box
  db = DBSCAN(eps=2.0, metric='euclidean', algorithm='auto', p=None).
      fit(coordinates)
  core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
  core_samples_mask[db.core_sample_indices_] = True
  labels = db.labels_
 current_clusters = len(set(labels))
 n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
 import matplotlib.pyplot as plt
 fig = plt.figure()

 if flag_hist == 1:
  ax = fig.add_subplot(121, projection ='3d')
  bx = fig.add_subplot(122)
 else:
```

```
95   ax = fig.add_subplot(111, projection='3d')
96  unique_labels = set(labels)
97  colors = plt.cm.Spectral(np.linspace(0, 1, len(unique_labels)))
98  for k, col in zip(unique_labels, colors):
99       if k == -1:
100
101           col = 'k'
102
103       class_member_mask = (labels == k)
104       xy = coordinates[class_member_mask & core_samples_mask]
105
106       ax.plot(xy[:, 0], xy[:, 1],xy[:,2], 'o', markerfacecolor=col,
107    markeredgecolor='k', markersize=14)
108       #print ("################ Inizio Gyration
              ####################")
109       #print ("XCoord: ")
110       #print (xy[:,0])
111       if (len(xy[:,0]) != 0) and (j>1000000/FRAME_COLLECTION):
112        xCm = (sum(xy[:,0])/len(xy[:,0]))
113        xDistCm = [x - xCm for x in xy[:,0]]
114        xDistCmSquared = [i ** 2 for i in xDistCm]
115
116       #print (X_dist_cm)
117       #print (X_dist_cm_squared)
118       #print ("X_cm: ")
119       #print (X_cm)
120       #print ("YCoord: ")
121       #print (xy[:,1])
122        yCm = (sum(xy[:,1])/len(xy[:,1]))
123        yDistCm = [y - yCm for y in xy[:,1]]
124        yDistCmSquared = [i ** 2 for i in yDistCm]
125       #print ("Y_cm: ")
126       #print (Y_cm)
127       #print ("ZCoord: ")
128       #print (xy[:,2])
129        zCm = (sum(xy[:,2])/len(xy[:,2]))
130        zDistCm = [z - zCm for z in xy[:,2]]
131        zDistCmSquared = [i ** 2 for i in zDistCm]
132        xyzDistCmSquared = [x+y+z for x,y,z in zip(xDistCmSquared,
              yDistCmSquared,zDistCmSquared)]
133        #xyzDistCmSquaredCorrected = [x*3/5 for x in xyzDistCmSquared]
134        xyzDistCmSquaredAv = (sum(xyzDistCmSquared[:]) / len(
              xyzDistCmSquared[:]))
135        gyrationRadius = sqrt(xyzDistCmSquaredAv)
136        #xyzDistCmRooted = [sqrt(x) for x in xyzDistCmSquared]
137        #gyrationRadius = (sum(xyzDistCmRooted)/len(xyzDistCmRooted))
138        if (len(xy[:,0]) != 0) and (gyrationRadius > 1):
139         clusterSize.append(round((len(xy[:,0])/ppo)))
140         clusterGyrRad.append(gyrationRadius)
141
```

```
142         #print (clusterSize)
143         #print (clusterGyrRad)
144          print (gyrationRadius)
145        #print (X_dist_cm_squared)
146        #print (Y_dist_cm_squared)
147        #print (Z_dist_cm_squared)
148        #print (XYZ_dist_cm_squared)
149        #print ("Y_cm: ")
150        #print (Y_cm)
151        #print ("################ Fine Gyration ####################")
152        times[j].beadHisto.append(len(xy[:,0]))
153        if times[j].beadHisto != 0:
154         times[j].chainHisto.append(round((len(xy[:,0]) / ppo)))
155        ax.plot(xy[:, 0], xy[:, 1],xy[:,2], 'o', markerfacecolor=col,
156     markeredgecolor='k', markersize=6)
157        check_status = check_status + (len(xy[:,0]))
158   if flag_hist == 1:
159    normalized_bin = len(times[j].chainHisto)
160    bx.hist(times[j].chainHisto, histtype='stepfilled')
161   CheckCluster.append(n_clusters_)
162   timecounter.append(j*FRAME_COLLECTION)
163   plt.ylim((0,30))
164   plt.xlim((0,30))
165   #plt.zlim((0,30))
166   plt.title('Estimated number of clusters: %d' % n_clusters_)
167   plt.savefig('timestep_'+str((1+j)*FRAME_COLLECTION)+'.png')
168   plt.close()
169   if (j >= (index-frameskip)):
170    fig_2 = plt.figure()
171    cx = fig_2.add_subplot(111)
172    cx.plot(timecounter,CheckCluster,'o',markersize=6,ls='--')
173    plt.title('#Cluster Vs Time')
174    plt.xlabel('Timestep')
175    plt.ylabel('#Cluster')
176    plt.savefig('cluster_'+str(j*500)+'.png')
177    plt.close()
178   fig_3 = plt.figure()
179   gx = fig_3.add_subplot(111)
180   gx.plot(clusterSize,clusterGyrRad,'o',markersize=5)
181   plt.title('Gyr Rad')
182   plt.xlabel('Size')
183   plt.ylabel('GyrRad')
184   plt.savefig('gyrRadCluster_'+str(j*500)+'.png')
185   plt.close()
186 from scipy.optimize import curve_fit
187 from pylab import *
188 clusterSize = np.asarray([(n) for n in clusterSize])
189
190 clusterGyrRad = np.asarray([(t) for t in clusterGyrRad])
191 m,b = polyfit(clusterSize,clusterGyrRad,1)
```

```
192 print (m)
193 print (b)
194 fig_5 = plt.figure()
195 plt.plot(clusterSize,m*clusterSize+b,'--k')
196 plt.plot(clusterSize,clusterGyrRad,'o',markersize=5)
197 plt.title(' Rad Log')
198 plt.xlabel('Size')
199 plt.ylabel('GyrRad')
200 plt.savefig('GyrRad.png')
201 plt.close()
202
203 clusterSize = np.asarray([mth.log(n) for n in clusterSize])
204
205 clusterGyrRad = np.asarray([mth.log(t) for t in clusterGyrRad])
206
207 from pylab import *
208
209 m,b = polyfit(clusterSize,clusterGyrRad,1)
210
211
212 #print(len(clusterSize))
213 #print(len(clusterGyrRad))
214 print (m)
215 print (b)
216 fig_4 = plt.figure()
217 plt.plot(clusterSize,m*clusterSize+b,'--k')
218 plt.plot(clusterSize,clusterGyrRad,'o',markersize=5)
219 plt.title('Gyr Rad Log')
220 plt.xlabel('Size')
221 plt.ylabel('GyrRad')
222 plt.savefig('LogGyrRad.png')
223 plt.close()
224 print ("End of the Cluster Analysis")
225 print ("Starting Histogram Averaging")
226 if (flag_ave == 1 ):
227  average_histograms_in_time(times, 1000000, 1200000,frameskip,
       FRAME_COLLECTION)
228 # average_histograms_in_time(times, 500000, 700000,frameskip,
     FRAME_COLLECTION)
229 # average_histograms_in_time(times, 850000, 1050000,frameskip,
     FRAME_COLLECTION)
230 print ("Check Your Results in the 'Results' Folder")
```

A.3 Fluent UDF - Laakkonen kernel

```
1  /* Subroutine for the solution of the Population Balance Equation with
       the */
2  /* Quadrature Method of Moments by using three nodes (N=3) and six
     moments */
3  /* from the moment of order zero to the moment of order five
                  */
4
5  /* The subroutine uses SIX user-defined scalars
                                */
6  /* S0 - moment of order zero
                                                          */
7  /* S1 - moment of order one
                                                           */
8  /* S2 - moment of order two
                                                           */
9  /* S3 - moment of order three
                                                         */
10 /* S4 - moment of order four
                                                           */
11 /* S5 - moment of order five
                                                           */
12
13 /* The subroutine uses 34 user-defined memories
                                */
14 /* M00 - size of bubble class 1 (abscissa 1)
                                  */
15 /* M01 - size of bubble class 2 (abscissa 2)
                                  */
16 /* M02 - size of bubble class 3 (abscissa 3)
                                  */
17 /* M03 - number density of bubble class 1 (weight 1)
                               */
18 /* M04 - number density of bubble class 2 (weight 2)
                               */
19 /* M05 - number density of bubble class 3 (weight 3)
                               */
20 /* M06 - source term for moment of order zero
                                */
21 /* M07 - source term for moment of order one
                                  */
22 /* M08 - source term for moment of order two
                                  */
23 /* M09 - source term for moment of order three
                                */
24 /* M10 - source term for moment of order four
                                */
25 /* M11 - source term for moment of order five
                                */
```

```
26 /* M12 - REAL moment of order zero                                    */
27 /* M13 - REAL moment of order one                                     */
28 /* M14 - REAL moment of order two                                     */
29 /* M15 - REAL moment of order three                                   */
30 /* M16 - REAL moment of order four                                    */
31 /* M17 - REAL moment of order five                                    */
32 /* M18 - average bubble diameter                                      */
33 /* M19 - average bubble diameter                                      */
34 /* M20 - debugging variable                                           */
35 /* M21 - debugging variable                                           */
36 /* M22 - debugging variable                                           */
37 /* M23 - debugging variable                                           */
38 /* M24 - debugging variable                                           */
39 /* M25 - debugging variable                                           */
40 /* M26 - debugging variable                                           */
41 /* M27 - debugging variable                                           */
42 /* M28 - debugging variable                                           */
43 /* M29 - debugging variable                                           */
44 /* M30 - debugging variable                                           */
45 /* M31 - check variable to track where moments are corrected          */
46 /* M32 - debugging variable                                           */
47 /* M33 - debugging variable                                           */
48 /* M34 - Turbulence integral Length scale                             */
49 /* M35 - alpha_x_visc                                                 */
50 #include "udf.h"
51 #include "mem.h"
52 #include "sg.h"
53 #include "sg_mem.h"
```

```
54 #include "sg_mphase.h"
55
56
57 /* Density of the gas (dispersed) phase, kg/m3 */
58 # define rho_g 946
59
60 /* Desnity of the liquid (continuous) phase, kg/m3 */
61 # define rho_c 998.2
62
63 /* Viscosity of the liquid (continuous) phase, kg/ms */
64 # define mu_c 0.032
65
66 /* Interfacial energy between continuous and disperse phases, N/m */
67 # define sigma 0.01160
68
69 /* Initial bubble diameter, m */
70 # define db0   3e-5
71
72 /* Costants appearing in the break-up kernel */
73 # define Cg   0.0035
74 # define Cx   0.23
75 # define Cp   1.4
76 # define beta_star 3
77 # define C1     3.68
78 # define C2     0.0775
79 # define C3     0.2
80 /*
81 # define C1     0.00481
82 # define C2     0.08
83 # define C3     0.0
84 */
85 /* Costant appearing in the daugther distribution function */
86 /* When C4 = 2.0 binary breakage is assumed to be the      */
87 /* dominant phenomenon                                     */
88 # define C4     2.0
89
90 /* Costants appearing in the coalescence kernel */
91
92 # define C7     0
93 # define C8     0
94 /*
95 # define C7     0.88
96 # define C8     9e15
97 */
98 /* definition of pi */
99 # define pigreco 3.14159265358979
100
101 /* Definition of the averaged diameter of the gas bubbles */
102
103
```

```
104 DEFINE_PROPERTY(bubble_diameter,c,t)
105 {
106
107    real diameter;
108
109
110  diameter = C_UDMI(c,t,18);
111
112
113  return diameter;
114
115 }
116
117
118
119 /* Definition of the moments source terms = rho(k)*S(k) */
120
121
122
123 DEFINE_SOURCE(p_source_m0, c, t, dS, eqn)
124 {
125
126  real source_m0;
127
128
129  source_m0 = rho_g*C_UDMI(c,t,6);
130
131
132  return source_m0;
133
134 }
135
136
137
138 DEFINE_SOURCE(p_source_m1, c, t, dS, eqn)
139 {
140
141  real source_m1;
142
143
144  source_m1 = rho_g*C_UDMI(c,t,7);
145
146
147  return source_m1;
148
149 }
150
151
152 DEFINE_SOURCE(p_source_m2, c, t, dS, eqn)
153 {
```

```
154
155  real source_m2;
156
157
158  source_m2 = rho_g*C_UDMI(c,t,8);
159
160
161  return source_m2;
162
163 }
164
165
166
167 DEFINE_SOURCE(p_source_m3, c, t, dS, eqn)
168 {
169
170  real source_m3;
171
172
173  source_m3 = rho_g*C_UDMI(c,t,9);
174
175
176  return source_m3;
177
178 }
179
180
181
182 DEFINE_SOURCE(p_source_m4, c, t, dS, eqn)
183 {
184
185  real source_m4;
186
187
188  source_m4 = rho_g*C_UDMI(c,t,10);
189
190
191  return source_m4;
192
193 }
194
195
196
197 DEFINE_SOURCE(p_source_m5, c, t, dS, eqn)
198 {
199
200  real source_m5;
201
202
203  source_m5 = rho_g*C_UDMI(c,t,11);
```

```
204
205
206   return source_m5;
207
208 }
209
210
211 /* Definition of the define adjust subroutine */
212
213 DEFINE_ADJUST(qmom_adj, domain)
214 {
215
216   cell_t c;
217
218   Thread *t;
219
220   int nt;
221
222
223           real mom[7];
224
225           real p1[8],p2[7],p3[6],p4[5],p5[4],p6[3],p7[2];
226
227   real alfa[7],d[4],e[4];
228
229           real birth[7],death[7],a[4],w[4];
230
231           real bi[4], a_b, E, A;
232
233   real beta;
234
235
236
237 /* Added variables for moments convexity check and correction*/
238
239
240   real d0[6],d1[5],d2[4],d3[3],d4[2],d5[1];
241
242   real bk[6][3];
243
244   real cos_quad_alfa[6];
245
246   int convexity_check,mom45_check,valid_set;
247
248   int k_star, conv_count1, conv_count2;
249
250   real lnck,min_mom_4,min_mom_5,min_mom_0,min_mom_1,min_mom_2,min_mom_3
          ,det_n,det_n1,det_n2,det_n3,det_n4,det_n5;
251
252   real old_mom[7],mom_a[7],mom_b[7],soglia_d2;
```

```
253
254  real sigma_var,log_mean,sigma_var_min;
255
256
257      int i,j;
258
259  int k;
260
261
262
263 /* Added variables for implementation of Lakkonen breakage and
       coalescence*/
264
265  real aj,bjb,djb,Bbb,Dbb,Bbc,Dbc,arg;
266
267  real arg_erf,a_erf,erf_app,epsilon;
268  real kappa,group_A,group_B,group_C,group_sqrt,group_sqrt_sum,L,
       alpha_x_visco,beta_mu;
269  real eff_coal,freq_coal,bjc,djc;
270
271  real x_gl[8],w_gl[8],t_gl,pre_int;
272
273
274  real breakage_1, breakage_2;
275
276
277 /* Added variables for eigenvalue eigenvector problem */
278
279
280          int m,l,iter;
281
282
283  real absa,absb;
284
285  real dd,g,r,SIGN,s,c1,p,f,b;
286
287  real z1[4],z2[4],z3[4];
288
289
290  thread_loop_c (t,domain)
291  {
292
293
294  Thread **pt = THREAD_SUB_THREADS(t);
295
296
297   begin_c_loop (c,t)
298  {
299
300           C_UDMI(c,t,33) = sigma;
```

```
301
302 /*  C_UDSI: user-defined scalar vector, cell value */
303 /*  C_VOF: cell volume fraction of pt(i) phase     */
304 /*  mom: vector used for the PD algorithm          */
305
306       if (C_UDSI(c,t,0)>0) mom[1] = C_UDSI(c,t,0)*C_VOF(c,pt[1]);
307
308       else mom[1] = 0.0;
309
310             if (C_UDSI(c,t,1)>0) mom[2] = C_UDSI(c,t,1)*C_VOF(c,pt
                  [1]);
311
312             else mom[2] = 0.0;
313
314           if (C_UDSI(c,t,2)>0) mom[3] = C_UDSI(c,t,2)*C_VOF(c,pt[1]);
315
316                 else mom[3] = 0.0;
317
318           if (C_UDSI(c,t,3)>0) mom[4] = C_UDSI(c,t,3)*C_VOF(c,pt[1]);
319
320                  else mom[4] = 0.0;
321
322                 if (C_UDSI(c,t,4)>0) mom[5] = C_UDSI(c,t,4)*C_VOF(c,
                      pt[1]);
323
324                 else mom[5] = 0.0;
325
326                 if (C_UDSI(c,t,5)>0) mom[6] = C_UDSI(c,t,5)*C_VOF(c,
                      pt[1]);
327
328                 else mom[6] = 0.0;
329
330 /* Algorithm for moment check and correction */
331
332     for (i=1;i<=6;i++)
333  {
334
335    old_mom[i]=mom[i];
336     }
337       C_UDMI(c,t,31)=0;
338
339  if ((mom[1]<=0.0) || (mom[2]<=0.0) || (mom[3]<=0.0) || (mom[4]<=0.0))
340 /* if mom[1] negative then a narrow distribution around the initial
     bubble size is assumed */
341    {
342        sigma_var=0.0001;
343        log_mean=log(db0);
344
345   mom[1]=exp(0.);
346
```

```
347   mom[2]=mom[1]*exp(log_mean+sigma_var/2.);
348
349       mom[3]=mom[1]*exp(2.*log_mean+4.*sigma_var/2.);
350
351       mom[4]=mom[1]*exp(3.*log_mean+9.*sigma_var/2.);
352
353                                   mom[5]=mom[1]*exp(4.*log_mean+16.*sigma_var
                                          /2.);
354
355          mom[6]=mom[1]*exp(5.*log_mean+25.*sigma_var/2.);
356
357       C_UDMI(c,t,31)=1;
358
359      }
360
361
362   soglia_d2=0.0;
363
364   sigma_var_min=0.0;
365
366
367  /* Check 1 convexty */
368
369    d0[0]=log(mom[1]);
370    d0[1]=log(mom[2]);
371    d0[2]=log(mom[3]);
372    d0[3]=log(mom[4]);
373     d0[4]=log(mom[5]);
374    d0[5]=log(mom[6]);
375    d1[0]=d0[1]-d0[0];
376    d1[1]=d0[2]-d0[1];
377    d1[2]=d0[3]-d0[2];
378    d1[3]=d0[4]-d0[3];
379    d1[4]=d0[5]-d0[4];
380    d2[0]=d1[1]-d1[0];
381    d2[1]=d1[2]-d1[1];
382    d2[2]=d1[3]-d1[2];
383    d2[3]=d1[4]-d1[3];
384    d3[0]=d2[1]-d2[0];
385    d3[1]=d2[2]-d2[1];
386    d3[2]=d2[3]-d2[2];
387    d4[0]=d3[1]-d3[0];
388             d4[1]=d3[2]-d3[1];
389             d5[0]=d4[1]-d4[0];
390             convexity_check=1;
391    conv_count1=0;
392    conv_count2=0;
393    for (i=0;i<=3;i++)  {
394        if ((d2[i])<soglia_d2) convexity_check=0;
395              }
```

```
396
397 /* End of test 1 */
398
399 /* Definition of bk */
400
401
402
403   bk[0][0]=-1.;
404
405   bk[0][1]=0.;
406
407     bk[0][2]=0.;
408
409
410          bk[1][0]=3.;
411
412     bk[1][1]=-1.;
413
414     bk[1][2]=0.;
415
416
417     bk[2][0]=-3.;
418
419     bk[2][1]=3.;
420
421     bk[2][2]=-1.;
422
423
424   bk[3][0]=1.;
425
426               bk[3][1]=-3.;
427
428     bk[3][2]=3.;
429
430
431   bk[4][0]=0.;
432
433    bk[4][1]=1.;
434
435   bk[4][2]=-3.;
436
437
438   bk[5][0]=0.;
439
440   bk[5][1]=0.;
441
442   bk[5][2]=1.;
443
444
445 /* Test 2 for moments of order four and five */
```

```
446
447       if ((mom[1]*mom[3]) != (mom[2]*mom[2]))
448       min_mom_4=(mom[1]*mom[4]*mom[4]-2*mom[3]*mom[2]*mom[4]+mom[3]*mom
              [3]*mom[3])/(mom[1]*mom[3]-mom[2]*mom[2]);
449
450   else
451      min_mom_4=mom[5];
452
453             if ((mom[2]*mom[4]) != (mom[3]*mom[3]))
454       min_mom_5=(mom[4]*(mom[4]*mom[4]-2*mom[3]*mom[5])+mom[2]*mom[5]*
              mom[5])/(mom[2]*mom[4]-mom[3]*mom[3]);
455
456   else
457      min_mom_5=mom[6];
458
459
460   mom45_check=1;
461
462
463      if (((mom[5]<min_mom_4)) || ((mom[6]<min_mom_5)))
464      mom45_check=0;
465
466 /* End test 2 */
467
468
469       while ((convexity_check==0) || (mom45_check==0)) {
470
471
472
473 /* Start correction after test 1 convexity */
474
475        k_star=0;
476
477
478      for (k=0;k<=5;k++) {
479
480            cos_quad_alfa[k]=pow(((d3[0]*bk[k][0]+d3[1]*bk[k][1]+d3[2]*bk
                  [k][2])/(pow(d3[0]*d3[0]+d3[1]*d3[1]+d3[2]*d3[2],0.5)*pow(
                  bk[k][0]*bk[k][0]+bk[k][1]*bk[k][1]+bk[k][2]*bk[k][2],0.5)
                  )),2.);
481             if (cos_quad_alfa[k] >= cos_quad_alfa[k_star])
482          k_star=k;
483           }
484
485           lnck=-(d3[0]*bk[k_star][0]+d3[1]*bk[k_star][1]+d3[2]*bk[k_star
                  ][2])/(bk[k_star][0]*bk[k_star][0]+bk[k_star][1]*bk[k_star
                  ][1]+bk[k_star][2]*bk[k_star][2]);
486
487
488 mom[k_star+1]=exp(lnck)*mom[k_star+1];
```

```
489
490
491     d0[0]=log(mom[1]);
492     d0[1]=log(mom[2]);
493     d0[2]=log(mom[3]);
494     d0[3]=log(mom[4]);
495     d0[4]=log(mom[5]);
496     d0[5]=log(mom[6]);
497     d1[0]=d0[1]-d0[0];
498     d1[1]=d0[2]-d0[1];
499     d1[2]=d0[3]-d0[2];
500     d1[3]=d0[4]-d0[3];
501     d1[4]=d0[5]-d0[4];
502     d2[0]=d1[1]-d1[0];
503     d2[1]=d1[2]-d1[1];
504     d2[2]=d1[3]-d1[2];
505     d2[3]=d1[4]-d1[3];
506     d3[0]=d2[1]-d2[0];
507             d3[1]=d2[2]-d2[1];
508             d3[2]=d2[3]-d2[2];
509             d4[0]=d3[1]-d3[0];
510     d4[1]=d3[2]-d3[1];
511     d5[0]=d4[1]-d4[0];
512             convexity_check=1;
513
514 /* End correction after test 1 convexity */
515
516       for (i=0;i<=3;i++) {
517        if ((d2[i])<soglia_d2)
518       convexity_check=0;
519       }
520
521 /* Start correction after check 2 of moments 4 and 5 */
522
523          if ((mom[1]*mom[3]) != (mom[2]*mom[2]))
524      min_mom_4=(mom[1]*mom[4]*mom[4]-2*mom[3]*mom[2]*mom[4]+mom[3]*mom
           [3]*mom[3])/(mom[1]*mom[3]-mom[2]*mom[2]);
525
526       else
527      min_mom_4=mom[5];
528
529     if ((mom[2]*mom[4]) != (mom[3]*mom[3]))
530             min_mom_5=(mom[4]*(mom[4]*mom[4]-2*mom[3]*mom[5])+mom[2]*
                 mom[5]*mom[5])/(mom[2]*mom[4]-mom[3]*mom[3]);
531
532     else
533      min_mom_5=mom[6];
534
535
536     mom45_check=1;
```

```
537
538     if (((mom[5]<min_mom_4)) || ((mom[6]<min_mom_5)))
539      mom45_check=0;
540
541 /* End correction after check 2 */
542
543       if (convexity_check == 0) {
544       conv_count1=conv_count1+1;
545       if (conv_count1>=10) {
546
547
548 /* Recover the initial moment values before convexity test */
549         for (i=1;i<=6;i++)  {
550          mom[i]=old_mom[i];
551                 }
552
553 /* Calculation of log-normal paramters starting from m0, m3, m2 */
554
555         i=2;
556
557     j=3;
558
559     sigma_var=((2./(j*j))*log(mom[j+1]/mom[1]) -(2./(i*j))*log(mom[i
        +1]/mom[1]))/(1.-((real)i/(real)j));
560
561     if (sigma_var<0)
562        sigma_var=sigma_var_min;
563         log_mean=(log(mom[i+1]/mom[1])-(i*i*sigma_var/2.))/i;
564
565              mom_a[1]=mom[1];
566
567       mom_a[2]=mom[1]*exp(log_mean+sigma_var/2.);
568
569                        mom_a[3]=mom[1]*exp(2.*log_mean+4.*sigma_var/2.);
570
571     mom_a[4]=mom[1]*exp(3.*log_mean+9.*sigma_var/2.);
572
573                        mom_a[5]=mom[1]*exp(4.*log_mean+16.*sigma_var/2.)
                             ;
574
575                      mom_a[6]=mom[1]*exp(5.*log_mean+25.*sigma_var/2.);
576
577
578 /* Calculation of log-normal paramters starting from m0, m3, m1 */
579
580       i=1;
581
582     j=3;
583                 sigma_var=((2./(j*j))*log(mom[j+1]/mom[1]) -(2./(i*j))*
                       log(mom[i+1]/mom[1]))/(1.-((real)i/(real)j));
```

```
584
585           if (sigma_var<0)
586          sigma_var=sigma_var_min;
587
588           log_mean=(log(mom[i+1]/mom[1])-(i*i*sigma_var/2.))/i;
589
590                  mom_b[1]=mom[1];
591
592                        mom_b[2]=mom[1]*exp(log_mean+sigma_var/2.);
593
594         mom_b[3]=mom[1]*exp(2.*log_mean+4.*sigma_var/2.);
595
596       mom_b[4]=mom[1]*exp(3.*log_mean+9.*sigma_var/2.);
597
598                        mom_b[5]=mom[1]*exp(4.*log_mean+16.*sigma_var/2.);
599
600                           mom_b[6]=mom[1]*exp(5.*log_mean+25.*sigma_var
                                /2.);
601
602
603  /* Assume the six moments equal to the average of those of the two
       distributions */
604
605                 mom[1]=mom[1];
606
607       mom[2]=(mom_a[2]+mom_b[2])/2.;
608
609       mom[3]=(mom_a[3]+mom_b[3])/2.;
610
611                        mom[4]=(mom_a[4]+mom_b[4])/2.;
612
613                        mom[5]=(mom_a[5]+mom_b[5])/2.;
614
615                        mom[6]=(mom_a[6]+mom_b[6])/2.;
616
617
618           convexity_check=1;
619
620       mom45_check=1;
621       }
622   /* end if conv_count>soglia */
623            }
624
625   /* end if convexity_check=0 */
626       else {
627      if (mom45_check==0) {
628       conv_count2=conv_count2+1;
629       if (conv_count2>=10) {
630  /* Calculation of log-normal paramters starting from m0, m3, m2 */
631          i=2;
```

```
632
633         j=3;
634
635         sigma_var=((2./(j*j))*log(mom[j+1]/mom[1])-(2./(i*j))*log(mom[i
                +1]/mom[1]))/(1.-((real)i/(real)j));
636
637         if (sigma_var<0)
638         sigma_var=sigma_var_min;
639
640         log_mean=(log(mom[i+1]/mom[1])-(i*i*sigma_var/2.))/i;
641
642         mom_a[1]=mom[1];
643
644         mom_a[2]=mom[1]*exp(log_mean+sigma_var/2.);
645
646         mom_a[3]=mom[1]*exp(2.*log_mean+4.*sigma_var/2.);
647
648         mom_a[4]=mom[1]*exp(3.*log_mean+9.*sigma_var/2.);
649
650         mom_a[5]=mom[1]*exp(4.*log_mean+16.*sigma_var/2.);
651
652         mom_a[6]=mom[1]*exp(5.*log_mean+25.*sigma_var/2.);
653
654 /* Calculation of log-normal paramters starting from m0, m3, m1 */
655
656         i=1;
657
658         j=3;
659
660         sigma_var=((2./(j*j))*log(mom[j+1]/mom[1])-(2./(i*j))*log(mom[i
                +1]/mom[1]))/(1.-((real)i/(real)j));
661
662         if (sigma_var<0)
663        sigma_var=sigma_var_min;
664
665         log_mean=(log(mom[i+1]/mom[1])-(i*i*sigma_var/2.))/i;
666
667         mom_b[1]=mom[1];
668
669         mom_b[2]=mom[1]*exp(log_mean+sigma_var/2.);
670
671         mom_b[3]=mom[1]*exp(2.*log_mean+4.*sigma_var/2.);
672
673         mom_b[4]=mom[1]*exp(3.*log_mean+9.*sigma_var/2.);
674
675         mom_b[5]=mom[1]*exp(4.*log_mean+16.*sigma_var/2.);
676
677         mom_b[6]=mom[1]*exp(5.*log_mean+25.*sigma_var/2.);
678
```

```
679 /* Assume the six moments equal to the average of those of the two
       distributions */
680
681
682         mom[1]=mom[1];
683
684         mom[2]=(mom_a[2]+mom_b[2])/2.;
685
686         mom[3]=(mom_a[3]+mom_b[3])/2.;
687
688         mom[4]=(mom_a[4]+mom_b[4])/2.;
689
690         mom[5]=(mom_a[5]+mom_b[5])/2.;
691
692         mom[6]=(mom_a[6]+mom_b[6])/2.;
693
694
695         mom45_check=1;
696
697       }
698 /* end if conv_count >= 50 */
699
700         } /* end mom45_check */
701
702   } /* end else convexity_check=0 */
703
704  } /* end while */
705
706 /* Gramian determinats
707
708      det_n=mom[1]*mom[3]*mom[5]-mom[1]*mom[4]*mom[4]-mom[2]*mom[2]*mom
           [5]+mom[2]*mom[3]*mom[4]+mom[2]*mom[3]*mom[4]-mom[3]*mom[3]*mom
           [3];
709      det_n1=mom[2]*mom[4]*mom[6]-mom[2]*mom[5]*mom[5]-mom[3]*mom[3]*mom
           [6]+mom[3]*mom[4]*mom[5]+mom[3]*mom[4]*mom[5]-mom[4]*mom[4]*mom
           [4];
710      det_n2=mom[1]*mom[3]-mom[2]*mom[2];
711      det_n3=mom[2]*mom[4]-mom[3]*mom[3];
712      det_n4=mom[2];
713      det_n5=mom[1];
714
715      valid_set=1;
716      if ((det_n<-1E-6) || (det_n1<-1E-6) || (det_n2<-1E-6) || (det_n3
           <-1E-6) || (det_n4<-1E-6) || (det_n5<-1E-6)) valid_set=0;
717
718 /* end */
719
720
721
722  C_UDMI(c,t,29) = 1.;
```

```
723
724
725  if (convexity_check==0)
726   C_UDMI(c,t,29) = -1.;
727
728
729
730 C_UDMI(c,t,30) = conv_count1;
731
732
733  C_UDMI(c,t,32) = conv_count2;
734
735
736 if ((C_VOF(c,pt[1])>1e-10))
737   C_UDSI(c,t,0)=mom[1]/C_VOF(c,pt[1]);
738
739  else
740    C_UDSI(c,t,0)=0.;
741
742 if ((C_VOF(c,pt[1])>1e-10))
743   C_UDSI(c,t,1)=mom[2]/C_VOF(c,pt[1]);
744
745  else
746   C_UDSI(c,t,1)=0.;
747
748
749 if ((C_VOF(c,pt[1])>1e-10))
750   C_UDSI(c,t,2)=mom[3]/C_VOF(c,pt[1]);
751
752  else
753   C_UDSI(c,t,2)=0.;
754
755  if ((C_VOF(c,pt[1])>1e-10))
756   C_UDSI(c,t,3)=mom[4]/C_VOF(c,pt[1]);
757
758  else
759   C_UDSI(c,t,3)=0.;
760
761  if ((C_VOF(c,pt[1])>1e-10))
762   C_UDSI(c,t,4)=mom[5]/C_VOF(c,pt[1]);
763
764  else
765   C_UDSI(c,t,4)=0.;
766
767
768  if ((C_VOF(c,pt[1])>1e-10))
769    C_UDSI(c,t,5)=mom[6]/C_VOF(c,pt[1]);
770
771          else
772   C_UDSI(c,t,5)=0.;
```

```
773
774
775
776 /* Product Difference algorithm */
777
778 p1[1]=1.0;
779 p1[2]=0.0;
780 p1[3]=0.0;
781 p1[4]=0.0;
782 p1[5]=0.0;
783 p1[6]=0.0;
784 p1[7]=0.0;
785
786  for (i=1;i<=6;i++)
787   {
788
789                 p2[i]=pow(-1,i-1)*mom[i];
790
791           }
792
793
794           for (i=1;i<=5;i++)
795  {
796
797                 p3[i]=p2[1]*p1[i+1]-p1[1]*p2[i+1];
798
799           }
800
801
802           for (i=1;i<=4;i++)
803  {
804
805                 p4[i]=p3[1]*p2[i+1]-p2[1]*p3[i+1];
806
807           }
808
809
810           for (i=1;i<=3;i++)
811  {
812
813                 p5[i]=p4[1]*p3[i+1]-p3[1]*p4[i+1];
814
815           }
816
817
818           for (i=1;i<=2;i++)
819  {
820
821                 p6[i]=p5[1]*p4[i+1]-p4[1]*p5[i+1];
822
```

```
823             }
824
825
826             p7[1]=p6[1]*p5[2]-p5[1]*p6[2];
827
828
829
830             alfa[1]=0.;
831
832
833             if (p2[1]*p1[1]!=0.)
834
835                     alfa[2]=p3[1]/(p2[1]*p1[1]);
836
837
838             if (p3[1]*p2[1]!=0.)
839
840                     alfa[3]=p4[1]/(p3[1]*p2[1]);
841
842
843             if (p4[1]*p3[1]!=0.)
844
845                     alfa[4]=p5[1]/(p4[1]*p3[1]);
846
847
848             if (p5[1]*p4[1]!=0.)
849
850                     alfa[5]=p6[1]/(p5[1]*p4[1]);
851
852
853             if (p6[1]*p5[1]!=0.)
854
855                     alfa[6]=p7[1]/(p6[1]*p5[1]);
856
857
858             for (i=1;i<=3;i++)
859     {
860
861                   d[i] = alfa[2*i]+alfa[2*i-1];
862
863             }
864
865
866             for (i=1;i<=2;i++)
867     {
868
869                   e[i] = pow(fabs(alfa[2*i+1]*alfa[2*i]),0.5);
870
871             }
872
```

```
873
874             e[3]=0.0;
875
876
877             z1[2]=z1[3]=0.0;
878
879             z2[1]=z2[3]=0.0;
880
881             z3[1]=z3[2]=0.0;
882
883             z1[1]=z2[2]=z3[3]=1.0;
884
885
886
887  /* Begin calculation eigenvalues and eigenvectors */
888
889
890             for (l=1;l<=3;l++) {
891
892    iter=0;
893
894    do {
895
896        for (m=l;m<=2;m++) {
897
898     dd=fabs(d[m])+fabs(d[m+1]);
899
900     if (fabs(e[m])+dd == dd)
901      break;
902         }
903
904
905           if (m != l) {
906
907
908    if (iter == 30)
909       break;
910     iter++;
911
912
913                    g=(d[l+1]-d[l])/(2.0*e[l]);
914
915
916              absa=fabs(g);
917
918                      absb=fabs(1.0);
919
920                      if (absa > absb)
921      r=absa*sqrt(1.0+pow(absb/absa,2));
922
```

```
923             else {
924
925                     if (absb==0)
926       r=0;
927
928                     else
929       r=absb*sqrt(1.0+pow(absa/absb,2));
930
931                 }
932
933
934                 if
935           (g<0) SIGN=-fabs(r);
936
937                 else
938
939                  SIGN = fabs(r);
940
941
942         g=d[m]-d[l]+e[l]/(g+SIGN);
943
944                 s=c1=1.0;
945
946                 p=0.0;
947
948
949                 for (i=m-1;i>=l;i--) {
950
951                   f=s*e[i];
952
953       b=c1*e[i];
954
955             absa=fabs(f);
956
957                           absb=fabs(g);
958
959                         if (absa > absb)
960        r=absa*sqrt(1.0+pow(absb/absa,2));
961
962                           else {
963
964                       if (absb==0)
965       r=0;
966
967                     else
968       r=absb*sqrt(1.0+pow(absa/absb,2));
969
970                       }
971
972                     e[i+1]=r;
```

```
973
974                     if (r == 0.0) {
975
976       d[i+1] -= p;
977
978               e[m]=0.0;
979
980                  break;
981
982             }
983
984
985             s=f/r;
986
987              c1=g/r;
988
989                     g=d[i+1]-p;
990
991                     r=(d[i]-g)*s+2.0*c1*b;
992
993                     d[i+1]=g+(p=s*r);
994
995                     g=c1*r-b;
996
997
998                          f=z1[i+1];
999
1000             z1[i+1]=s*z1[i]+c1*f;
1001
1002             z1[i]=c1*z1[i]-s*f;
1003
1004
1005             f=z2[i+1];
1006
1007             z2[i+1]=s*z2[i]+c1*f;
1008
1009             z2[i]=c1*z2[i]-s*f;
1010
1011
1012             f=z3[i+1];
1013
1014             z3[i+1]=s*z3[i]+c1*f;
1015
1016             z3[i]=c1*z3[i]-s*f;
1017
1018
1019                  }
1020
1021               if (r == 0.0 && i >= l)
1022       continue;
```

```
1023
1024        d[l] -= p;
1025
1026        e[l]=g;
1027
1028        e[m]=0.0;
1029
1030
1031        }
1032
1033    }
1034   while (m != l);
1035
1036    }
1037
1038
1039      if (fabs(d[1])>0.) {
1040
1041      if (fabs(d[2])/fabs(d[1])>1.0e3)
1042    d[2] = d[1];
1043
1044               if (fabs(d[3])/fabs(d[1])>1.0e3) d[3] = d[1];
1045
1046       }
1047
1048
1049  /*  End calculation eigenvalues and eigenvectors */
1050  /*  Assign nodes a(i) and weights w(i) to M0 - M5 */
1051
1052
1053
1054      C_UDMI(c,t,0) = a[1] = fabs(d[1]);
1055      C_UDMI(c,t,1) = a[2] = fabs(d[2]);
1056      C_UDMI(c,t,2) = a[3] = fabs(d[3]);
1057      C_UDMI(c,t,3) = w[1] = pow(z1[1],2)*mom[1];
1058      C_UDMI(c,t,4) = w[2] = pow(z1[2],2)*mom[1];
1059      C_UDMI(c,t,5) = w[3] = pow(z1[3],2)*mom[1];
1060      a_erf=(8./(3.*pigreco))*(pigreco-3.)/(4.-pigreco);
1061      pre_int=(9.+16.5*C4+9.*pow(C4,2.)+1.5*pow(C4,3.));
1062
1063      epsilon=C_D(c,t);
1064      kappa=C_K(c,t);
1065
1066  /* Nodes and weight for integration of the daugther distribution
        function with Gauss-Legendre */
1067
1068
1069   x_gl[1] = -0.949107912342759;
1070
1071   x_gl[2] = -0.741531185599394;
```

```
1072
1073  x_gl[3] = -0.405845151377397;
1074
1075  x_gl[4] = 0.0;
1076
1077  x_gl[5] = -x_gl[3];
1078
1079  x_gl[6] = -x_gl[2];
1080
1081  x_gl[7] = -x_gl[1];
1082
1083  w_gl[1] = 0.129484966168869;
1084
1085  w_gl[2] = 0.279705391489276;
1086
1087  w_gl[3] = 0.381830050505118;
1088
1089  w_gl[4] = 0.417959183673469;
1090
1091  w_gl[5] = w_gl[3];
1092
1093  w_gl[6] = w_gl[2];
1094
1095  w_gl[7] = w_gl[1];
1096
1097
1098
1099 /* Calculation of alpha(i) with nodes and weights */
1100
1101
1102
1103  C_UDMI(c,t,25) = pigreco/6.*w[1]*a[1]*a[1]*a[1];
1104  C_UDMI(c,t,26) = pigreco/6.*w[2]*a[2]*a[2]*a[2];
1105  C_UDMI(c,t,27) = pigreco/6.*w[3]*a[3]*a[3]*a[3];
1106   C_UDMI(c,t,28) = C_UDMI(c,t,25)+C_UDMI(c,t,26)+C_UDMI(c,t,27)+1.-
        C_VOF(c,pt[1]);
1107
1108
1109     for (k=1;k<=6;k++)
1110  {
1111
1112   birth[k] = 0.;
1113
1114  death[k] = 0.;
1115
1116
1117  Bbb=0.;
1118
1119   Dbb=0.;
1120
```

```
1121  Bbc=0.;
1122
1123   Dbc=0.;
1124
1125
1126   for (i=1;i<=3;i++)
1127 {
1128
1129
1130 /* controlla che alpha-p(i) sia >1E-3
1131
1132    if (C_UDMI(c,t,(24+i))<0.001) {
1133
1134    a[i]=0.;
1135
1136    w[i]=0.;
1137
1138  }
1139
1140 /* Verify that a[i] exists */
1141
1142    if(a[i]>0.0)  {
1143
1144 /* Breakage with kernel from Laakkonen */
1145     bjb=0.;
1146
1147 /* Integration Gauss-Legendre 6 nodes  */
1148
1149
1150     /*
1151 for (j=1;j<=7;j++)  {
1152
1153      aj=(x_gl[j]+1.)/2.*a[i];
1154
1155      t_gl=aj/a[i];
1156
1157     bjb=bjb+pre_int*a[i]*pow((t_gl*a[i]),(k-1))*((pow(t_gl,2.))/(a[i])
          )*(pow(t_gl,6.))*pow((1-pow(t_gl,3.)),C4)*w_gl[j];
1158
1159      }
1160
1161 */
1162      bjb = (3240*pow(a[i],(k-1))) / ((k+9-1)*(k+12-1)*(k+15-1));
1163
1164 /* Approx of error function */
1165
1166
1167
1168 /* Laakkonen Kernel */
1169
```

```
1170      arg=((C2*(sigma/(rho_c*pow(epsilon,2./3.)*pow(a[i],5./3.))))+(C3*(
              mu_c/(pow((rho_c*rho_g),0.5)*pow(epsilon,1./3.)*pow(a[i],4./3.)
              ))));
1171
1172      arg_erf=pow(arg,0.5);
1173
1174      erf_app=pow((1.-exp(-pow(arg_erf,2.)*((4./pigreco+a_erf*pow(
              arg_erf,2.))/(1.+a_erf*pow(arg_erf,2.))))),0.5);
1175
1176      L = pow((2*kappa/3),3/2)/epsilon;
1177
1178      beta_mu = log(2)*pow(Cx,-5/3)/(beta_star*Cp);
1179            group_A = (beta_mu*pow(Cx,5/3)*mu_c)/(rho_c*pow(epsilon,1/3)*
                  pow(L,(1/3))*a[i]);
1180            group_B = (beta_mu*pow(Cx,5/3)*mu_c)/(rho_c*pow(epsilon,1/3)*
                  pow(L,(1/3))*a[i]);
1181            group_C = (4*pow(Cx,5/3)*sigma)/(rho_c*pow(L,2/3)*a[i]*pow(
                  epsilon,2/3));
1182            group_sqrt = pow((pow(group_B,2)+group_C),0.5);
1183            group_sqrt_sum = 2*pow((group_A + group_sqrt),-1);
1184      alpha_x_visco = 3*log(group_sqrt_sum)/log(L/a[i]);
1185
1186      Bbb=Bbb+(C1*pow(epsilon,1./3.)*(1-erf_app))*w[i]*bjb;
1187      Dbb=Dbb+(C1*pow(epsilon,1./3.)*(1-erf_app))*w[i]*pow(a[i],(k-1));
1188
1189
1190  /* Source terms */
1191
1192
1193
1194  /*Tavlarides Kernel */
1195  /*      breakage_1 = -C2*sigma/(rho_c*pow(epsilon,2./3.)*pow(a[i
          ],5./3.));
1196        breakage_2 =  C1*pow(epsilon,1./3)/(pow(a[i],2./3.));
1197
1198        Bbb=Bbb+breakage_2*exp(breakage_1)*w[i]*bjb;
1199
1200        Dbb=Dbb+breakage_2*exp(breakage_1)*w[i]*pow(a[i],(k-1));
1201
1202   */
1203  /* Coalescence with kernel from Laakkonen */
1204
1205
1206      bjc=0.;
1207
1208      djc=0.;
1209
1210
1211    for (j=1;j<=3;j++)  {
1212
```

```
1213  if (a[i]*a[j] != 0.) {
1214
1215        eff_coal=exp((-C8)*(mu_c*rho_c*epsilon)/(sigma*sigma)*pow((a[i]*
                a[j]/(a[i]+a[j])),4.));
1216
1217        freq_coal=C7*pow(epsilon,(1./3.))*pow((a[i]+a[j]),2.)*pow((pow(a
                [i],(2./3.))+pow(a[j],(2./3.))),0.5)*eff_coal;
1218
1219        bjc=bjc+w[j]*pow((pow(a[i],3.)+pow(a[j],3.)),(k-1)/3.)*freq_coal
                ;
1220
1221        djc=djc+freq_coal*w[j];
1222
1223       }
1224
1225       }
1226
1227
1228     Bbc=Bbc+bjc*w[i];
1229
1230      Dbc=Dbc+djc*pow(a[i],(k-1))*w[i];
1231
1232    }
1233
1234  }
1235    birth[k] = birth[k] + Bbb + Bbc*0.5;
1236    death[k] = death[k] + Dbb + Dbc;
1237
1238  }
1239
1240
1241  /* Calculate moments source terms and assign to M6 - M11*/
1242  C_UDMI(c,t,35) = alpha_x_visco;
1243
1244  C_UDMI(c,t,34) = pow((2*kappa/3),3/2)/epsilon;
1245  C_UDMI(c,t,6)  = birth[1]-death[1];
1246  C_UDMI(c,t,7)  = birth[2]-death[2];
1247  C_UDMI(c,t,8)  = birth[3]-death[3];
1248  C_UDMI(c,t,9)  = birth[4]-death[4];
1249  C_UDMI(c,t,10) = birth[5]-death[5];
1250  C_UDMI(c,t,11) = birth[6]-death[6];
1251  /* Calculate real moments m(k) = phi(k)*alfa(d) and assign to M12 -
         M18*/
1252
1253  C_UDMI(c,t,12) = C_UDSI(c,t,0)*C_VOF(c,pt[1]);
1254  C_UDMI(c,t,13) = C_UDSI(c,t,1)*C_VOF(c,pt[1]);
1255  C_UDMI(c,t,14) = C_UDSI(c,t,2)*C_VOF(c,pt[1]);
1256  C_UDMI(c,t,15) = C_UDSI(c,t,3)*C_VOF(c,pt[1]);
1257  C_UDMI(c,t,16) = C_UDSI(c,t,4)*C_VOF(c,pt[1]);
1258  C_UDMI(c,t,17) = C_UDSI(c,t,5)*C_VOF(c,pt[1]);
```

```
1259
1260 /* Calculation of Sauter diameter */
1261
1262 if (mom[3]>1.0e-3)
1263  C_UDMI(c,t,18) = mom[4]/mom[3];
1264
1265 else
1266  C_UDMI(c,t,18) = db0;
1267
1268 if (C_UDMI(c,t,18)>2.0e-3)
1269  C_UDMI(c,t,20) = E = 0.77*pow(C_UDMI(c,t,18)*1000,-0.316) ;
1270
1271 else
1272  C_UDMI(c,t,20) = E = 0.77*pow(2.0,-0.316);
1273
1274 C_UDMI(c,t,21) = A = 1.0/2.0/pow(E,2./3.)*( 1 + pow(E,2) / 2.0 / pow
       (1-pow(E,2),0.5) * log((1+pow(1-pow(E,2),0.5))/(1-pow(1-pow(E,2)
       ,0.5))) );
1275       }
1276  end_c_loop (c,t)
1277 }
1278
1279 }
```

Appendix B

PBE Algorithm

The roots of the system presented in Chapter 3 can obtained by solving the following system:

$$\xi \begin{vmatrix} P_0(\xi) \\ P_1(\xi) \\ P_2(\xi) \\ \vdots \\ P_{N-2}(\xi) \\ P_{N-1}(\xi) \end{vmatrix} = \begin{vmatrix} a_0 & 1 & & & & \\ b_1 & a_1 & 1 & & & \\ & b_2 & a_2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & a_{2N-2} & 1 \\ & & & & b_{N-1} & a_{N-1} \end{vmatrix} \begin{vmatrix} P_0(\xi) \\ P_1(\xi) \\ P_2(\xi) \\ \vdots \\ P_{N-2}(\xi) \\ P_{N-1}(\xi) \end{vmatrix} + \begin{vmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ P_N\xi \end{vmatrix} \tag{B.1}$$

Where the roots of $P_N(\xi)$ are the eigenvalues of the tridiagonal matrix, that must be made symmetric to transform the ill-conditioned problem into well-conditioned. Weights can be obtained as follows:

$$w_\alpha = m_0 \phi_{\alpha 1}^2 \tag{B.2}$$

where $\phi_{\alpha 1}$ is the first component of the αth of the eigenvector ϕ_α of the Jacobi matrix:

$$\mathbf{J} = \begin{vmatrix} a_0 & \sqrt{b_1} & & & & \\ \sqrt{b_1} & a_1 & \sqrt{b_2} & & & \\ & & \ddots & \ddots & & \\ & & & & a_{N-2} & \sqrt{b_{N-1}} \\ & & & & \sqrt{b_{N-1}} & a_{N-1} \end{vmatrix} \tag{B.3}$$

B.1 Product Difference Algorithm

One way to calculate the coefficients introduced in Chapter 3 is the Product Difference Algorithm (PD), introduced by Gordon. A P matrix is constructed (N = 2):

$$P = \begin{vmatrix} 1 & m_0 & m_1 & m_0 m_2 - (m_1)^2 & m_0(m_3 m_1 - (m_2)^2) \\ 0 & -m_1 & -m_2 & -(m_0 m_3 - m_2 m_1) & \\ 0 & m_2 & m_3 & & \\ 0 & -m_3 & & & \\ 0 & & & & \end{vmatrix} \tag{B.4}$$

where each of the terms of the matrix can be obtained in the following way:

$$P_{\alpha,1} = \delta_{\alpha,1}, \tag{B.5}$$

$$P_{\alpha,2} = (-1)^{\alpha-1} m_{\alpha-1}, \tag{B.6}$$

$$P_{\alpha,\beta} = P_{1,\beta-1} P_{\alpha+1,\beta-2} - P_{1,\beta-2} P_{\alpha+1,\beta-1} \tag{B.7}$$

B.2 Wheeler Algorithm

The Wheeler algorithm can be used as an alternative to the PD algorithm. A new set of basis functions is used to represent the orthogonal polynomials. Coefficients are obtained from modified moments:

$$v_k = \int_{\sigma_\xi} \pi_k(\xi) n(\xi) d(\xi) \tag{B.8}$$

with $k = 0,1,..,2N-1$.. Jacobi matrix coefficients can be obtained through intermediate quantities:

$$\sigma_{\alpha,\beta} = \int_{\sigma_\xi} n(\xi) \pi_\alpha(\xi) \pi_\beta(\xi) d(\xi) \tag{B.9}$$

and the coefficients result:

$$a_\alpha = a'_\alpha - \frac{\sigma_{\alpha-1,\alpha}}{\sigma_{\alpha-1,\alpha-1}} + \frac{\sigma_{\alpha,\alpha+1}}{\sigma_{\alpha,\alpha}} \tag{B.10}$$

$$b_\alpha = \frac{\sigma_{\alpha,\alpha}}{\sigma_{\alpha-1,\alpha-1}}, \tag{B.11}$$

and more details can be found in Marchisio D. and Fox. R. (Marchisio and Fox, 2013).

*

Bibliography

Alexandridis, P. (2000), "Block copolymers", *Current Opinion in Colloid and Interface Science*, 5, 5-6, pp. 312-313.

— (1997), "Poly(ethylene oxide)/poly(propylene oxide) block copolymer surfactants", *Current Opinion in Colloid and Interface Science*, 2, 5, pp. 478-489.

Alexandridis, P., U. Olsson, and B. Lindman (1998), "A record nine different phases (four cubic, two hexagonal, and one lamellar lyotropic liquid crystalline and two micellar solutions) in a ternary isothermal system of an amphiphilic block copolymer and selective solvents (water and oil)", *Langmuir*, 14, 10, pp. 2627-2638.

— (1995), "Self-Assembly of Amphiphilic Block Copolymers: The (EO)13(PO)30(EO)13–Water–p-Xylene System", *Macromolecules*, 28, 23, pp. 7700-7710.

Allen, M. P. and D. J. Tildesley (2017), *Computer simulation of liquids: Second edition*, pp. 1-626.

Almgren, M., W. Brown, and S. Hvidt (1995), "Self-aggregation and phase behavior of poly(ethylene oxide)-poly(propylene oxide)-poly(ethylene oxide) block copolymers in aqueous solution", *Colloid Polymer Science*, 273, 1, pp. 2-15.

Álvarez-Ramírez, J. G., V. V. A. Fernández, E. R. Macías, Y. Rharbi, P. Taboada, R. Gámez-Corrales, J. E. Puig, and J. F. A. Soltero (2009), "Phase behavior of the Pluronic P103/water system in the dilute and semi-dilute regimes", *Journal of colloid and interface science*, 333, 2, pp. 655-662.

Andersson, B., R. Andersson, L. Håkansson, M. Mortensen, R. Sudiyo, and B. Van Wachem (2011), "Computational fluid dynamics for engineers", in Computational Fluid Dynamics for Engineers, vol. 9781107018952, pp. 1-189.

Aubin, J., D. F. Fletcher, and C. Xuereb (2004), "Modeling turbulent flow in stirred tanks with CFD: The influence of the modeling approach, turbulence model and numerical scheme", *Experimental Thermal and Fluid Science*, 28, 5, pp. 431-445.

Aydin, F., X. Chu, G. Uppaladadium, D. Devore, R. Goyal, N. S. Murthy, Z. Zhang, J. Kohn, and M. Dutt (2016), "Self-Assembly and Critical Aggregation Concentration Measurements of ABA Triblock Copolymers with Varying B Block Types: Model Development, Prediction, and Validation", *Journal of Physical Chemistry B*, 120, 15, pp. 3666-3676.

Baldyga, J., J. R. Bourne, and R. V. Gholap (1995), "The influence of viscosity on mixing in jet reactors", *Chemical Engineering Science*, 50, 12, pp. 1877-1880.

Baldyga, J. and W. Podgórska (1998), "Drop break-up in intermittent turbulence: maximum stable and transient sizes of drops", *Canadian Journal of Chemical Engineering*, 76, 3, pp. 456-470.

Binks, B. P. (2002), "Particles as surfactants - Similarities and differences", *Current Opinion in Colloid and Interface Science*, 7, 1-2, pp. 21-41.

Binks, Bernard P. (1998), *Modern aspects of emulsion science*, Royal Society of Chemistry.

Bird, R.B., W. E. Stewart, and E. Lightfoot (2002), "Transport Phenomena", *Jhon Wiley and Sons*.

Boek, E. S., P. V. Coveney, H. N. W. Lekkerkerker, and P. van der Schoot (1997), "Simulating the rheology of dense colloidal suspensions using dissipative particle dynamics", *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, 55, 3, pp. 3124-3133.

Brennen, C. E. (2013), "Fundamentals of multiphase flow", in Fundamentals of Multiphase Flow, vol. 9780521848046, pp. 1-345.

Buffo, A., J. De Bona, M. Vanni, and D. L. Marchisio (2016), "Simplified volume-averaged models for liquid–liquid dispersions: Correct derivation and comparison with other approaches", *Chemical Engineering Science*, 153, pp. 382-393.

Cao, X., G. Xu, Y. Li, and Z. Zhang (2005), "Aggregation of poly(ethylene oxide) - Poly(propylene oxide) block copolymers in aqueous solution: DPD simulation study", *Journal of Physical Chemistry A*, 109, 45, pp. 10418-10423.

Carraher C. E., Jr. (2017), "Introduction to polymer chemistry fourth edition", in Introduction to Polymer Chemistry Fourth Edition, pp. 1-560.

Chatterjee, A. (2007), "Modification to Lees-Edwards periodic boundary condition for dissipative particle dynamics simulation with high dissipation rates", *Molecular Simulation*, 33, 15, pp. 1233-1236.

Cheng, F., X. Guan, H. Cao, T. Su, J. Cao, Y. Chen, M. Cai, B. He, Z. Gu, and X. Luo (2015), "Characteristic of core materials in polymeric micelles effect on their micellar properties studied by experimental and dpd simulation methods", *International journal of pharmaceutics*, 492, 1-2, pp. 152-160.

Chiarotti, G.F., F. Fermi, M.P. Tosi, and S. F. Edwards (1990), "TOpics in statistical mechanics of disordered systems", *Proc. Int. Sched. Phys.* P. 837.

Coroneo, M., G. Montante, A. Paglianti, and F. Magelli (2011), "CFD prediction of fluid flow and mixing in stirred tanks: Numerical issues about the RANS simulations", *Computers and Chemical Engineering*, 35, 10, pp. 1959-1968.

Coulaloglou, C. A. and L. L. Tavlarides (1977), "Description of interaction processes in agitated liquid-liquid dispersions", *Chemical Engineering Science*, 32, 11, pp. 1289-1297.

Cundall, P. A. and O. D. L. Strack (1979), "A discrete numerical model for granular assemblies", *Geotechnique*, 29, 1, Cited By :9082, pp. 47-65.

Di Pasquale, N., T. Hudson, and M. Icardi (2019), "Systematic derivation of hybrid coarse-grained models", *Physical Review E*, 99, 1.

Droghetti, H., I. Pagonabarraga, P. Carbone, P. Asinari, and D. Marchisio (2018), "Dissipative particle dynamics simulations of tri-block co-polymer and water: Phase diagram validation and microstructure identification", *Journal of Chemical Physics*, 149, 18.

Espanol, P. (2004), "Statistical Mechanics of Coarse-Graining", *Lecture Notes in Physics*, pp. 2256, 2256.

Español, P. (1995), "Hydrodynamics from dissipative particle dynamics", *Physical Review E*, 52, 2, pp. 1734-1742.

Español, P. and M. Revenga (2003), "Smoothed dissipative particle dynamics", *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 67, 2 2, pp. 267051-2670512.

Español, P. and P. B. Warren (2017), "Perspective: Dissipative particle dynamics", *Journal of Chemical Physics*, 146, 15.

Ester, M., H. Kriegel, J. Sander, and X. Xu (1996), "A density-based algorithm for discovering clusters in large spatial databases with noise", *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pp. 226-231.

Fedosov, D. A., B. Caswell, and G. Em Karniadakis (2008), "Reverse poiseuille flow: the numerical viscometer", in *AIP Conference Proceedings*, vol. 1027.

Fedosov, D. A., G. E. Karniadakis, and B. Caswell (2010), "Steady shear rheometry of dissipative particle dynamics models of polymer fluids in reverse Poiseuille flow", *Journal of Chemical Physics*, 132, 14.

Flory, P. J. (1941), "Thermodynamics of high polymer solutions", *The Journal of chemical physics*, 9, 8, pp. 660-661.

Frenkel, D. and B. Smit (1996), "Understanding molecular simulation: From algorithms to applications", in Understanding molecular simulation: From algorithms to applications.

Gao, Z., D. Li, A. Buffo, W. Podgórska, and D. L. Marchisio (2016), "Simulation of droplet breakage in turbulent liquid-liquid dispersions with CFD-PBM: Comparison of breakage kernels", *Chemical Engineering Science*, 142, pp. 277-288.

Gentile, L., M. A. Behrens, S. Balog, K. Mortensen, G. A. Ranieri, and U. Olsson (2014), "Dynamic phase diagram of a nonionic surfactant lamellar phase", *Journal of Physical Chemistry B*, 118, 13, pp. 3622-3629.

Gombosi, T. (1994), "Gaskinetic Theory", *Cambridge University Press*.

Groot, R. D. and T. J. Madden (1998), "Dynamic simulation of diblock copolymer microphase separation", *Journal of Chemical Physics*, 108, 20, pp. 8713-8724.

Groot, R. D. and K. L. Rabone (2001), "Mesoscopic simulation of cell membrane damage, morphology change and rupture by nonionic surfactants", *Biophysical journal*, 81, 2, pp. 725-736.

Groot, R. D. and P. B. Warren (1997), "Dissipative particle dynamics: Bridging the gap between atomistic and mesoscopic simulation", *Journal of Chemical Physics*, 107, 11, pp. 4423-4435.

Guo, Q. (2016), "Polymer morphology: Principles, characterization, and processing", in Polymer Morphology: Principles, Characterization, and Processing, pp. 1-445.

Hall, S., M. Cooke, A. El-Hamouz, and A. J. Kowalski (2011), "Droplet break-up by in-line Silverson rotor-stator mixer", *Chemical Engineering Science*, 66, 10, pp. 2068-2079.

Hall, S., M. Cooke, A. W. Pacek, A. J. Kowalski, and D. Rothman (2011), "Scaling up of silverson rotor-stator mixers", *Canadian Journal of Chemical Engineering*, 89, 5, pp. 1040-1050.

Hammouda, B. (2010), "SANS from Pluronic P85 in d-water", *European Polymer Journal*, 46, 12, pp. 2275-2281.

EL-Hamouz, A., M. Cooke, A. Kowalski, and P. Sharratt (2009), "Dispersion of silicone oil in water surfactant solution: Effect of impeller speed, oil viscosity and addition point on drop size distribution", *Chemical Engineering and Processing: Process Intensification*, 48, 2, pp. 633-642.

Holmqvist, P., P. Alexandridis, and B. Lindman (1998), "Modification of the microstructure in block copolymer-water-"oil" systems by varying the copolymer composition and the "oil" type: Small-angle X-ray scattering and deuterium-NMR investigation", *Journal of Physical Chemistry B*, 102, 7, pp. 1149-1158.

Hoogerbrugge, P. J. and J. M. V. A. Koelman (1992), "Simulating microscopic hydrodynamic phenomena with dissipative particle dynamics", *EPL*, 19, 3, pp. 155-160.

Huggins, M. L. (1942), "Some properties of solutions of long-chain compounds", *Journal of Physical Chemistry*, 46, 1, pp. 151-158.

Ishii, M., K. Mishima, I. Katsoka, and G. Kocamustafaogullari (1982), "Two-fluid model and importance of the interfacial area in two-phase flow analysis." English.

Jain, S. and F. S. Bates (2003a), "On the origins of morphological complexity in block copolymer surfactants", *Science*, 300, 5618, pp. 460-464.

— (2003b), "On the origins of morphological complexity in block copolymer surfactants", *Science*, 300, 5618, pp. 460-464.

James, J., M. Cooke, A. Kowalski, and T. L. Rodgers (2017), "Scale-up of batch rotor-stator mixers. Part 2—Mixing and emulsification", *Chemical Engineering Research and Design*, 124, pp. 321-329.

James, J., M. Cooke, L. Trinh, R. Hou, P. Martin, A. Kowalski, and T. L. Rodgers (2017), "Scale-up of batch rotor–stator mixers. Part 1—power constants", *Chemical Engineering Research and Design*, 124, pp. 313-320.

Jansen, K. M. B., W. G. M. Agterof, and J. Mellema (2001), "Viscosity of surfactant stabilized emulsions", *Journal of Rheology*, 45, 6, pp. 1359-1371.

Janssen, J. M. H. and H. E. H. Meijer (1993), "Droplet breakup mechanisms: Stepwise equilibrium versus transient dispersion", *Journal of Rheology*, 37, 4, pp. 597-608.

Konno, M., M. Aoki, and S. Saito (1983), "Scale effect on breakup process in liquid-liquid agitated tanks", *Journal of Chemical Engineering of Japan*, 16, 4, pp. 312-319.

Kowalski, A. J., M. Cooke, and S. Hall (2011), "Expression for turbulent power draw of an in-line Silverson high shear mixer", *Chemical Engineering Science*, 66, 3, pp. 241-249.

Kremer, K. and G. S. Grest (1990), "Dynamics of entangled linear polymer melts: A molecular-dynamics simulation", *The Journal of chemical physics*, 92, 8, pp. 5057-5086.

Kubo, R. (1957), "Statistical'Mechanical Theory of Irreversible Processes. I. General Theory and Simple Applications to Magnetic and Conduction Problems", *Journal of the Physical Society of Japan*, 12, 6, pp. 570-586.

Laakkonen, M., P. Moilanen, V. Alopaeus, and J. Aittamaa (2007), "Modelling local bubble size distributions in agitated vessels", *Chemical Engineering Science*, 62, 3, pp. 721-740.

Lagisetty, J. S., P. K. Das, R. Kumar, and K. S. Gandhi (1986), "Breakage of viscous and non-Newtonian drops in stirred dispersions", *Chemical Engineering Science*, 41, 1, pp. 65-72.

Langevin, P. (1908), "On the Theory of Brownian Motion", *C. R. Acad. Sci. Paris*. Pp. 530-533.

Larson, Ronald G. (1999), "The Structure and Rheology of Complex Fluids", in pp. 1-665.

Lavino, A. D., N. Di Pasquale, P. Carbone, A. A. Barresi, and D. L. Marchisio (2015), "Simulation of macromolecule self-assembly in solution: A multiscale approach", in *AIP Conference Proceedings*, vol. 1695.

Leaf, B. (1972), "Derivation of the Boltzmann equation from the Liouville equation", *Physica*, 59, 2, pp. 206-227.

Lees, A. W. and S. F. Edwards (1972), "The computer study of transport processes under extreme conditions", *Journal of Physics C: Solid State Physics*, 5, 15, pp. 1921-1928.

Lennard-Jones, J. E. (1924), "On the determination of molecular fields", *Proc. R. Soc. Lond. A*, 106, pp. 463-477.

Li, D., Z. Gao, A. Buffo, W. Podgorska, and D. L. Marchisio (2017), "Droplet breakage and coalescence in liquid–liquid dispersions: Comparison of different kernels with EQMOM and QMOM", *AIChE Journal*, 63, 6, pp. 2293-2311.

Li, Y., H. Zhang, M. Bao, and Q. Chen (2012), "Aggregation Behavior of Surfactants with Different Molecular Structures in Aqueous Solution: DPD Simulation Study", *Journal of Dispersion Science and Technology*, 33, 10, pp. 1437-1443.

Lince, F., D. L. Marchisio, and A. A. Barresi (2008), "Strategies to control the particle size distribution of poly-ε-caprolactone nanoparticles for pharmaceutical applications", *Journal of colloid and interface science*, 322, 2, pp. 505-515.

M., Doi (2013), "Soft Matter Physics", pp. 1-272.

Malkin, A. Y. and A. I. Isayev (2011), "Rheology: Concepts, methods, and applications: Second edition", in Rheology: Concepts, Methods, and Applications: Second Edition, pp. 1-473.

Marchisio, D. L. and R. O. Fox (2013), *Computational models for polydisperse particulate and multiphase systems*, Cambridge University Press, pp. 1-508.

Marsh, C. A., G. Backx, and M. H. Ernst (1997), "Static and dynamic properties of dissipative particle dynamics", *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, 56, 2, pp. 1676-1691.

Meneveau, C. and K. R. Sreenivasan (1991), "The multifractal nature of turbulent energy dissipation", *Journal of Fluid Mechanics*, 224, pp. 429-484.

Morsi, S. A. and A. J. Alexander (1972), "An investigation of particle trajectories in Two-Phase flow systems", *J. Fluid Mech.* 55, pp. 193-208.

Moshfegh, A. and A. Jabbarzadeh (2015), "Modified Lees-Edwards boundary condition for dissipative particle dynamics: Hydrodynamics and temperature at high shear rates", *Molecular Simulation*, 41, 15, pp. 1264-1277.

Moulik, S. P. and B. K. Paul (1998), "Structure, dynamics and transport properties of micro emulsions", *Advances in Colloid and Interface Science*, 78, 2, pp. 99-195.

Nere, N. K., A. W. Patwardhan, and J. B. Joshi (2003), "Liquid-phase mixing in stirred vessels: Turbulent flow regime", *Industrial and Engineering Chemistry Research*, 42, 12, pp. 2661-2698.

Newby, G. E., I. W. Hamley, S. M. King, C. M. Martin, and N. J. Terrill (2009), "Structure, rheology and shear alignment of Pluronic block copolymer mixtures", *Journal of colloid and interface science*, 329, 1, pp. 54-61.

Nicolaides, D. (2001), "Mesoscale modelling", *Molecular Simulation*, 26, 1, pp. 51-72.

Osher, S. and R. P. Fedkiw (2001), "Level Set Methods: An Overview and Some Recent Results", *Journal of Computational Physics*, 169, 2, pp. 463-502.

Pasquino, R., H. Droghetti, P. Carbone, S. Mirzaagha, N. Grizzuti, and D. Marchisio (2019), "An experimental rheological phase diagram of a tri-block co-polymer in water validated against dissipative particle dynamics simulations", *Soft Matter*, 15, 6, pp. 1396-1404.

Pelesko, J. A. (2007), "Self assembly: The science of things that put themselves together", in Self Assembly: The Science of Things that Put Themselves Together, pp. 1-308.

Plimpton, S. (1995), "Fast parallel algorithms for short-range molecular dynamics", *Journal of Computational Physics*, 117, 1, pp. 1-19.

Prhashanna, A., S. A. Khan, and S. B. Chen (2016), "Micelle morphology and chain conformation of triblock copolymers under shear: LA-DPD study", *Colloids and Surfaces A: Physicochemical and Engineering Aspects*, 506, pp. 457-466.

Ramakrishna, D. (2000), "Population Balances: Theory and Applications to Particulate Systems in Engineering", *Academic Press*, p. 384.

Rehage, H. and H. Hoffmann (1988), "Rheological properties of viscoelastic surfactant systems", *Journal of Physical Chemistry*, 92, 16, pp. 4712-4719.

Sahu, A. K., P. Kumar, A. W. Patwardhan, and J. B. Joshi (1999), "CFD modelling and mixing in stirred tanks", *Chemical Engineering Science*, 54, 13-14, pp. 2285-2293.

Seaton, M. A., R. L. Anderson, S. Metz, and W. Smith (2013), "DL-MESO: Highly scalable mesoscale simulations", *Molecular Simulation*, 39, 10, pp. 796-821.

Shiller, L. and A. Naumann (1935), "A drag coefficient correlation", *Zeitschrift des Vereins Deutscher Ingenieure*, 77, pp. 318-320.

Smoluchowski, M. (1916), "Drei Vortrage uber Diffusion, Brownsche Bewegung und Koagulation von Kolloidteilchen", *Physik. Zeit.* 17, pp. 557-585.

Soddemann, T., B. Dünweg, and K. Kremer (2003), "Dissipative particle dynamics: A useful thermostat for equilibrium and nonequilibrium molecular dynamics simulations", *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 68, 4 2, pp. 467021-467028.

Sollich, P., F. Lequeux, P. Hébraud, and M. E. Cates (1997), "Rheology of soft glassy materials", *Physical Review Letters*, 78, 10, Cited By :771, pp. 2020-2023.

Sommerfeld, M. and S. Decker (2004), "State of the art and future trends in CFD simulation of stirred vessel hydrodynamics", *Chemical Engineering and Technology*, 27, 3, pp. 215-224.

Song, X., S. Zhao, S. Fang, Y. Ma, and M. Duan (2016), "Mesoscopic Simulations of Adsorption and Association of PEO-PPO-PEO Triblock Copolymers on a Hydrophobic Surface: From Mushroom Hemisphere to Rectangle Brush", *Langmuir*, 32, 44, pp. 11375-11385.

Sun, N., Y. Li, D. Wang, M. Bao, and L. Tong (2013), "Mesoscopic simulation studies on the self-assembly of Pluronic at oil/water interface", *Acta Chimica Sinica*, 71, 2, pp. 186-192.

Sundararajan, P. R. (2016), "Physical Aspects of Polymer Self-Assembly", in Physical Aspects of Polymer Self-Assembly, pp. 1-368.

Tadros, T. F. (2010), "Rheology of Dispersions: Principles and Applications", in Rheology of Dispersions: Principles and Applications.

Townsend, B., F. Peyronel, N. Callaghan-Patrachar, B. Quinn, A. G. Marangoni, and D. A. Pink (2017), "Shear-induced aggregation or disaggregation in edible oils: Models, computer simulation, and USAXS measurements", *Journal of Applied Physics*, 122.

Tryggvason, G., B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, and Y. -. Jan (2001), "A Front-Tracking Method for the Computations of Multiphase Flow", *Journal of Computational Physics*, 169, 2, pp. 708-759.

Valente, I., E. Celasco, D. L. Marchisio, and A. A. Barresi (2012), "Nanoprecipitation in confined impinging jets mixers: Production, characterization and scale-up of pegylated nanospheres and nanocapsules for pharmaceutical use", *Chemical Engineering Science*, 77, pp. 217-227.

Walstra, P. (1996), *Encyclopedia of Emulsion Technology*, P. Becher, Dekker.

— (1993), "Principles of emulsion formation", *Chemical Engineering Science*, 48, 2, pp. 333-349.

Walstra, P. and I. Smulders (1997), *Food Colloids: Proteins, Lipids and Polysaccharides*, The Royal Society of Chemistry, Cambridge.

Wang, S. -. (2017), "Nonlinear polymer rheology: Macroscopic phenomenology and molecular foundation", in Nonlinear Polymer Rheology: Macroscopic Phenomenology and Molecular Foundation, pp. 1-427.

Youssry, M., F. Asaro, L. Coppola, L. Gentile, and I. Nicotera (2010), "Solution microstructures of the micellar phase of Pluronic L64/SDS/water system", *Journal of colloid and interface science*, 342, 2, pp. 348-353.

Zhou, D., P. Alexandridis, and A. Khan (1996), "Self-assembly in a mixture of two poly(ethylene oxide)-b-poly(propylene oxide)-b-poly(ethylene oxide) copolymers in water", *Journal of colloid and interface science*, 183, 2, pp. 339-350.

This Ph.D. thesis has been typeset by means of the TEX-system facilities. The typesetting engine was LuaLATEX. The document class was `toptesi`, by Claudio Beccari, with option `tipotesi=scudo`. This class is available in every up-to-date and complete TEX-system installation.